RM0487
Reference manual

STM32U3 series Arm®-based 32-bit MCUs

Introduction

This reference manual complements the datasheets of the STM32U3 series microcontrollers, providing information required for application and in particular for software development. It pertains to the superset of feature sets available on STM32U3 series microcontrollers.

For feature set, ordering information, and mechanical and electrical characteristics of a particular STM32U3 series device, refer to its corresponding datasheet.

For information on the Arm® Cortex-M33 core, refer to the *Cortex-M33 technical reference manual*.

The STM32U3 series microcontrollers include ST state-of-the-art patented technology.

Related documents

- *Cortex-M33 Technical Reference Manual*, available from: http://infocenter.arm.com
- PM0223 programming manual for Cortex-M33 core(a)
- STM32U3 series datasheets(a)
- STM32U3 series errata sheets(a)
- AN2606 application note on booting STM32 MCUs(a)

a. Available on STMicroelectronics website *www.st.com*

Contents

1 Documentation conventions 87

1.1 General information 87

1.2 List of abbreviations for registers 87

1.3 Register reset value 87

1.4 Glossary 88

1.5 Availability of peripherals 88

2 Memory and bus architecture 89

2.1 System architecture 89

2.1.1 Fast C-bus 90

2.1.2 Slow C-bus 90

2.1.3 S-bus 90

2.1.4 GPDMA-bus 91

2.1.5 SDMMC1 controller DMA bus 91

2.1.6 Bus matrix 91

2.1.7 AHB/APB bridges 91

2.2 Arm TrustZone security architecture 91

2.2.1 Default Arm TrustZone security state 92

2.2.2 Arm TrustZone peripheral classification 93

2.3 Memory organization 97

2.3.1 Introduction 97

2.3.2 Memory map and register boundary addresses 98

2.3.3 Embedded SRAMs 102

2.3.4 Flash memory overview 103

3 System security 104

3.1 Key security features 104

3.2 Secure install 105

3.3 Secure boot 105

3.3.1 Unique boot entry and BOOT_LOCK 106

3.3.2 Immutable root of trust in system flash memory 106

3.4 Secure update 106

3.5 Resource isolation using TrustZone 107

3.5.1 TrustZone security architecture . . . 107
3.5.2 Armv8-M security extension of Cortex-M33 . . . 108
3.5.3 Memory and peripheral allocation using IDAU/SAU . . . 108
3.5.4 Memory and peripheral allocation using GTZC . . . 110
3.5.5 Managing security in TrustZone-aware peripherals . . . 113
3.5.6 Activating TrustZone security . . . 120
3.5.7 Deactivating TrustZone security . . . 121
3.6 Other resource isolations . . . 121
3.6.1 Temporal isolation using secure hide protection (HDP) . . . 121
3.6.2 Resource isolation using Cortex privileged mode . . . 122
3.7 Secure execution . . . 125
3.7.1 Memory protection unit (MPU) . . . 126
3.7.2 Embedded flash memory write protection . . . 126
3.7.3 Tamper detection and response . . . 126
3.8 Secure storage . . . 127
3.8.1 Hardware secret key management . . . 128
3.8.2 Unique ID . . . 129
3.9 Cryptographic engines . . . 129
3.9.1 Cryptographic engines features . . . 129
3.9.2 Secure AES co-processor (SAES) . . . 130
3.10 Product life-cycle . . . 131
3.10.1 Life-cycle management with readout protection (RDP) . . . 132
3.10.2 Recommended option-byte settings . . . 134
3.11 Access controlled debug . . . 135
3.11.1 Debug protection with readout protection (RDP) . . . 135
3.12 Software intellectual property protection and collaborative development 135
3.12.1 Software intellectual property protection with RDP . . . 136
3.12.2 Other software intellectual property protections . . . 137

4 Boot modes . . . 138
4.1 Embedded bootloader . . . 140
4.2 Embedded root security services (RSS) . . . 140

5 Global TrustZone controller (GTZC) . . . 141
5.0.1 GTZC introduction . . . 141
5.0.2 GTZC main features . . . 141

5.0.3 GTZC implementation . . . 143
5.1 GTZC functional description . . . 144
5.1.1 GTZC block diagram . . . 144
5.1.2 Illegal access definition . . . 145
5.1.3 TrustZone security controller (TZSC) . . . 146
5.1.4 Memory protection controller - block based (MPCBB) . . . 146
5.1.5 TrustZone illegal access controller (TZIC) . . . 146
5.1.6 Power-on/reset state . . . 147
5.1.7 GTZC interrupts . . . 147
5.2 GTZC1 TZSC registers . . . 147
5.2.1 GTZC1 TZSC control register (GTZC1_TZSC_CR) . . . 147
5.2.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1) . . . 148
5.2.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2) . . . 150
5.2.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3) . . . 152
5.2.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1) . . . 154
5.2.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2) . . . 156
5.2.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3) . . . 157
5.2.8 GTZC1 TZSC register map . . . 159
5.3 GTZC1 TZIC registers . . . 160
5.3.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1) . . . 160
5.3.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2) . . . 162
5.3.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3) . . . 163
5.3.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4) . . . 165
5.3.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1) . . . 166
5.3.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2) . . . 168
5.3.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3) . . . 170
5.3.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4) . . . 171
5.3.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1) . . . 173
5.3.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2) . . . 175
5.3.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3) . . . 176
5.3.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4) . . . 178
5.3.13 GTZC1 TZIC register map . . . 179

5.4 GTZC1 MPCBB1 registers 181
5.4.1 GTZC1 SRAM1 MPCBB control register (GTZC1_MPCBB1_CR) . . . 181
5.4.2 GTZC1 SRAM1 MPCBB configuration lock register 1 (GTZC1_MPCBB1_CFGLOCKR1) 182
5.4.3 GTZC1 SRAM1 MPCBB security configuration for super-block x register (GTZC1_MPCBB1_SECCFGRx) 182
5.4.4 GTZC1 SRAM1 MPCBB privileged configuration for super-block x register (GTZC1_MPCBB1_PRIVCFGRx) 183
5.4.5 GTZC1 MPCBB1 register map 183
5.5 GTZC1 MPCBB2 registers 184
5.5.1 GTZC1 SRAM2 MPCBB control register (GTZC1_MPCBB2_CR) . . . 184
5.5.2 GTZC1 SRAM2 MPCBB configuration lock register 1 (GTZC1_MPCBB2_CFGLOCKR1) 185
5.5.3 GTZC1 SRAM2 MPCBB security configuration for super-block x register (GTZC1_MPCBB2_SECCFGRx) 185
5.5.4 GTZC1 SRAM2 MPCBB privileged configuration for super-block x register (GTZC1_MPCBB2_PRIVCFGRx) 186
5.5.5 GTZC1 MPCBB2 register map 186

6 RAM configuration controller (RAMCFG) 187
6.1 RAMCFG introduction 187
6.2 RAMCFG main features 187
6.3 RAMCFG functional description 187
6.3.1 Internal SRAMs features 187
6.3.2 Parity error detection (SRAM2) 188
6.3.3 Write protection (SRAM2) 188
6.3.4 Software erase 188
6.4 RAMCFG in low-power modes 189
6.5 RAMCFG interrupts 189
6.6 RAMCFG registers 190
6.6.1 RAMCFG memory 1 control register (RAMCFG_M1CR) 190
6.6.2 RAMCFG memory 1 interrupt status register (RAMCFG_M1ISR) . . . 190
6.6.3 RAMCFG memory 2 control register (RAMCFG_M2CR) 191
6.6.4 RAMCFG memory 2 interrupt enable register (RAMCFG_M2IER) . . . 191
6.6.5 RAMCFG memory 2 interrupt status register (RAMCFG_M2ISR) . . . 192
6.6.6 RAMCFG memory 2 parity error address register (RAMCFG_M2PEAR) 192
6.6.7 RAMCFG memory 2 interrupt clear register (RAMCFG_M2ICR) 193

6.6.8 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) . . . 193
6.6.9 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2) . . . 194
6.6.10 RAMCFG memory 2 parity key register (RAMCFG_M2PARKEYR) . . 194
6.6.11 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) . . . 194
6.6.12 RAMCFG register map . . . 195

7 Embedded flash memory (FLASH) . . . 197

7.1 FLASH introduction . . . 197
7.2 FLASH main features . . . 197
7.3 FLASH functional description . . . 198
7.3.1 Flash memory organization . . . 198
7.3.2 Error code correction (ECC) . . . 199
7.3.3 Read access latency . . . 200
7.3.4 Bank power-down mode . . . 201
7.3.5 Flash memory program and erase operations . . . 202
7.3.6 Flash main memory erase sequences . . . 203
7.3.7 Flash memory programming sequences . . . 204
7.3.8 Flash memory errors flags . . . 206
7.3.9 Read-while-write (RWW) . . . 209
7.3.10 Power-down during flash programming or erase operation . . . 209
7.3.11 Reset during flash programming or erase operation . . . 209
7.4 FLASH option bytes . . . 210
7.4.1 Option-byte description . . . 210
7.4.2 Option-byte programming . . . 211
7.5 FLASH TrustZone security and privilege protections . . . 213
7.5.1 TrustZone security protection . . . 213
7.5.2 Watermark-based secure flash memory area protection . . . 214
7.5.3 Secure hide protection (HDP) . . . 215
7.5.4 Secure hide protection extension (HDP extension) . . . 216
7.5.5 Block-based secure flash memory area protection . . . 219
7.5.6 Flash security attribute state . . . 219
7.5.7 Block-based privileged flash memory area protection . . . 220
7.5.8 Flash memory register privileged and unprivileged modes . . . 220
7.5.9 Flash memory bank attributes in case of bank swap . . . 220

7.6 FLASH memory protection 222
7.6.1 Write protection (WRP) 222
7.6.2 Readout protection (RDP) 223
7.7 System memory 232
7.7.1 RSS user functions 232
7.8 Summary of flash memory and flash memory registers access control . 234
7.9 FLASH interrupts 237
7.10 FLASH registers 238
7.10.1 FLASH access control register (FLASH_ACR) 238
7.10.2 FLASH nonsecure key register (FLASH_KEYR) 239
7.10.3 FLASH secure key register (FLASH_SKEYR) 240
7.10.4 FLASH option key register (FLASH_OPTKEYR) 240
7.10.5 FLASH bank 1 power-down key register (FLASH_PDKEY1R) 241
7.10.6 FLASH bank 2 power-down key register (FLASH_PDKEY2R) 241
7.10.7 FLASH nonsecure status register (FLASH_SR) 242
7.10.8 FLASH secure status register (FLASH_SSR) 243
7.10.9 FLASH nonsecure control register (FLASH_CR) 245
7.10.10 FLASH secure control register (FLASH_SCR) 247
7.10.11 FLASH ECC register (FLASH_ECCCORR) 248
7.10.12 FLASH ECC detection register (FLASH_ECCDETR) 249
7.10.13 FLASH operation status register (FLASH_OPSR) 250
7.10.14 FLASH option register (FLASH_OPTR) 251
7.10.15 FLASH nonsecure boot address 0 register (FLASH_BOOT0R) 253
7.10.16 FLASH nonsecure boot address 1 register (FLASH_BOOT1R) 254
7.10.17 FLASH secure boot address 0 register (FLASH_SBOOT0R) 255
7.10.18 FLASH secure watermark1 register 1 (FLASH_SECWM1R1) 256
7.10.19 FLASH secure watermark1 register 2 (FLASH_SECWM1R2) 257
7.10.20 FLASH WRP1 area A address register (FLASH_WRP1AR) 258
7.10.21 FLASH WRP1 area B address register (FLASH_WRP1BR) 259
7.10.22 FLASH secure watermark2 register 1 (FLASH_SECWM2R1) 260
7.10.23 FLASH secure watermark2 register 2 (FLASH_SECWM2R2) 261
7.10.24 FLASH WPR2 area A address register (FLASH_WRP2AR) 262
7.10.25 FLASH WPR2 area B address register (FLASH_WRP2BR) 263
7.10.26 FLASH secure block based bank 1 register x (FLASH_SECBB1Rx) . 264
7.10.27 FLASH secure block based bank 2 register x (FLASH_SECBB2Rx) . 264

7.10.28 FLASH secure HDP control register (FLASH_SECHDPCR) 265
7.10.29 FLASH privilege configuration register (FLASH_PRIVCFGR) 266
7.10.30 FLASH HDP extension register (FLASH_SECHDPEXTR) 266
7.10.31 FLASH privilege block based bank 1 register x (FLASH_PRIVBB1Rx) 267
7.10.32 FLASH privilege block based bank 2 register x (FLASH_PRIVBB2Rx) 268
7.10.33 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1) 268
7.10.34 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2) 269
7.10.35 FLASH OEM1 key register 3 (FLASH_OEM1KEYR3) 269
7.10.36 FLASH OEM1 key register 4 (FLASH_OEM1KEYR4) 270
7.10.37 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1) 270
7.10.38 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2) 271
7.10.39 FLASH OEM2 key register 3 (FLASH_OEM2KEYR3) 271
7.10.40 FLASH OEM2 key register 4 (FLASH_OEM2KEYR4) 272
7.10.41 FLASH OEM key status register (FLASH_OEMKEYSR) 272
7.10.42 FLASH register map 273

8 Instruction cache (ICACHE) 277

8.1 ICACHE introduction 277
8.2 ICACHE main features 277
8.3 ICACHE implementation 278
8.4 ICACHE functional description 278
8.4.1 ICACHE block diagram 279
8.4.2 ICACHE reset and clocks 279
8.4.3 ICACHE TAG memory 280
8.4.4 Direct-mapped ICACHE (1-way cache) 281
8.4.5 ICACHE enable 282
8.4.6 Cacheable and noncacheable traffic 282
8.4.7 Address remapping 283
8.4.8 Cacheable accesses 285
8.4.9 Dual-master cache 286
8.4.10 ICACHE security 286
8.4.11 ICACHE maintenance 286
8.4.12 ICACHE performance monitoring 287
8.4.13 ICACHE boot 287
8.5 ICACHE low-power modes 287

8.6 ICACHE error management and interrupts 288
8.7 ICACHE registers 288
8.7.1 ICACHE control register (ICACHE_CR) 288
8.7.2 ICACHE status register (ICACHE_SR) 289
8.7.3 ICACHE interrupt enable register (ICACHE_IER) 290
8.7.4 ICACHE flag clear register (ICACHE_FCR) 290
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) 291
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) 291
8.7.7 ICACHE region x configuration register (ICACHE_CRRx) 291
8.7.8 ICACHE register map 293

9 Power control (PWR) 294
9.1 PWR introduction 294
9.2 PWR main features 294
9.3 PWR functional description 294
9.3.1 PWR pins and internal signals 294
9.3.2 PWR power supplies and supply domains 296
9.3.3 PWR system supply voltage regulation 300
9.3.4 PWR power supply supervision 302
9.3.5 PWR power management 305
9.3.6 PWR security and privileged protection 328
9.4 PWR interrupts 330
9.5 PWR registers 331
9.5.1 PWR control register 1 (PWR_CR1) 331
9.5.2 PWR control register 2 (PWR_CR2) 333
9.5.3 PWR control register 3 (PWR_CR3) 334
9.5.4 PWR voltage scaling register (PWR_VOSR) 335
9.5.5 PWR supply voltage monitoring control register (PWR_SVMCR) 336
9.5.6 PWR wake-up control register 1 (PWR_WUCR1) 338
9.5.7 PWR wake-up control register 2 (PWR_WUCR2) 338
9.5.8 PWR wake-up control register 3 (PWR_WUCR3) 339
9.5.9 PWR backup domain control register (PWR_BDCR) 340
9.5.10 PWR disable backup domain register (PWR_DBPR) 340
9.5.11 PWR security configuration register (PWR_SECCFGR) 341
9.5.12 PWR privilege control register (PWR_PRIVCFGR) 342
9.5.13 PWR status register (PWR_SR) 342

9.5.14 PWR supply voltage monitoring status register (PWR_SVMSR) 343
9.5.15 PWR wake-up status register (PWR_WUSR) 344
9.5.16 PWR wake-up status clear register (PWR_WUSCR) 344
9.5.17 PWR apply pull configuration register (PWR_APCR) 345
9.5.18 PWR port A pull-up control register (PWR_PUCRA) 345
9.5.19 PWR port A pull-down control register (PWR_PDCRA) 346
9.5.20 PWR port B pull-down control register (PWR_PDCRB) 347
9.5.21 PWR port x pull-up control register (PWR_PUCRx) (x = B to E) 348
9.5.22 PWR port x pull-down control register (PWR_PDCRx) (x = C to E) . . 349
9.5.23 PWR port G pull-up control register (PWR_PUCRG) 349
9.5.24 PWR port G pull-down control register (PWR_PDCRG) 350
9.5.25 PWR port H pull-up control register (PWR_PUCRH) 351
9.5.26 PWR port H pull-down control register (PWR_PDCRH) 351
9.5.27 PWR I3C pull-up control register 1 (PWR_I3CPUCR1) 352
9.5.28 PWR I3C pull-up control register 2 (PWR_I3CPUCR2) 353
9.5.29 PWR register map 354

10 Reset and clock control (RCC) 357

10.1 RCC introduction 357
10.2 RCC functional description 357
10.2.1 RCC pins and internal signals 357
10.2.2 RCC reset functional description 357
10.2.3 RCC clock functional description 359
10.2.4 RCC security and privilege functional description 374
10.3 RCC in low-power modes 377
10.4 RCC interrupts 378
10.5 RCC registers 380
10.5.1 RCC clock control register (RCC_CR) 380
10.5.2 RCC internal clock source calibration register 1 (RCC_ICSCR1) 383
10.5.3 RCC internal clock source calibration register 2 (RCC_ICSCR2) 386
10.5.4 RCC internal clock source calibration register 3 (RCC_ICSCR3) 387
10.5.5 RCC clock recovery RC register (RCC_CRRCR) 387
10.5.6 RCC clock configuration register 1 (RCC_CFGR1) 388
10.5.7 RCC clock configuration register 2 (RCC_CFGR2) 390
10.5.8 RCC clock configuration register 3 (RCC_CFGR3) 391
10.5.9 RCC clock configuration register 4 (RCC_CFGR4) 392
10.5.10 RCC clock interrupt enable register (RCC_CIER) 393

10.5.11 RCC clock interrupt flag register (RCC_CIFR) 394
10.5.12 RCC clock interrupt clear register (RCC_CICR) 396
10.5.13 RCC AHB1 peripheral reset register 1 (RCC_AHB1RSTR1) 397
10.5.14 RCC AHB2 peripheral reset register 1 (RCC_AHB2RSTR1) 398
10.5.15 RCC AHB2 peripheral reset register 2 (RCC_AHB2RSTR2) 399
10.5.16 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) 400
10.5.17 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) 401
10.5.18 RCC APB2 peripheral reset register (RCC_APB2RSTR) 402
10.5.19 RCC APB3 peripheral reset register (RCC_APB3RSTR) 403
10.5.20 RCC AHB1 peripheral clock enable register 1 (RCC_AHB1ENR1) . . . 403
10.5.21 RCC AHB2 peripheral clock enable register 1 (RCC_AHB2ENR1) . . . 405
10.5.22 RCC AHB2 peripheral clock enable register 2 (RCC_AHB2ENR2) . . . 406
10.5.23 RCC AHB1 peripheral clock enable register 2 (RCC_AHB1ENR2) . . . 406
10.5.24 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1) . . . 407
10.5.25 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2) . . . 409
10.5.26 RCC APB2 peripheral clock enable register (RCC_APB2ENR) 409
10.5.27 RCC APB3 peripheral clock enable register (RCC_APB3ENR) 410
10.5.28 RCC AHB1 peripheral clock enable in Sleep mode register (RCC_AHB1SLPENR1) 411
10.5.29 RCC AHB2 peripheral clock enable in Sleep mode register 1 (RCC_AHB2SLPENR1) 412
10.5.30 RCC AHB2 peripheral clock enable in Sleep mode register 2 (RCC_AHB2SLPENR2) 414
10.5.31 RCC AHB1 peripheral clock enable in Sleep mode register 2 (RCC_AHB1SLPENR2) 415
10.5.32 RCC APB1 peripheral clock enable in Sleep mode register 1 (RCC_APB1SLPENR1) 415
10.5.33 RCC APB1 peripheral clock enable in Sleep mode register 2 (RCC_APB1SLPENR2) 417
10.5.34 RCC APB2 peripheral clock enable in Sleep mode register (RCC_APB2SLPENR) 418
10.5.35 RCC APB3 peripheral clock enable in Sleep mode register (RCC_APB3SLPENR) 419
10.5.36 RCC AHB1 peripheral clock enable in Stop mode register (RCC_AHB1STPENR1) 420
10.5.37 RCC AHB2 peripheral clock enable in Stop mode register 1 (RCC_AHB2STPENR1) 421
10.5.38 RCC APB1 peripheral clock enable in Stop mode register 1 (RCC_APB1STPENR1) 422
10.5.39 RCC APB1 peripheral clock enable in Stop mode register 2 (RCC_APB1STPENR2) 423

10.5.40 RCC APB2 peripheral clock enable in Stop mode register (RCC_APB2STPENR) . . . 424
10.5.41 RCC APB3 peripheral clock enable in Stop mode register (RCC_APB3STPENR) . . . 425
10.5.42 RCC peripheral independent clock configuration register 1 (RCC_CCIPR1) . . . 426
10.5.43 RCC peripheral independent clock configuration register 2 (RCC_CCIPR2) . . . 428
10.5.44 RCC peripheral independent clock configuration register 3 (RCC_CCIPR3) . . . 430
10.5.45 RCC backup domain control register (RCC_BDCR) . . . 431
10.5.46 RCC control/status register (RCC_CSR) . . . 433
10.5.47 RCC secure configuration register (RCC_SECCFGR) . . . 435
10.5.48 RCC privilege configuration register (RCC_PRIVCFGR) . . . 436
10.5.49 RCC register map . . . 437

11 Clock recovery system (CRS) . . . 443

11.1 CRS introduction . . . 443
11.2 CRS main features . . . 443
11.3 CRS implementation . . . 443
11.4 CRS functional description . . . 444
11.4.1 CRS block diagram . . . 444
11.4.2 CRS internal signals . . . 444
11.4.3 Synchronization input . . . 445
11.4.4 Frequency error measurement . . . 445
11.4.5 Frequency error evaluation and automatic trimming . . . 446
11.4.6 CRS initialization and configuration . . . 447
11.5 CRS in low-power modes . . . 448
11.6 CRS interrupts . . . 448
11.7 CRS registers . . . 448
11.7.1 CRS control register (CRS_CR) . . . 448
11.7.2 CRS configuration register (CRS_CFGR) . . . 449
11.7.3 CRS interrupt and status register (CRS_ISR) . . . 450
11.7.4 CRS interrupt flag clear register (CRS_ICR) . . . 452
11.7.5 CRS register map . . . 453

12 General-purpose I/Os (GPIO) . . . 454

12.1 Introduction . . . 454

12.2 GPIO main features 454
12.3 GPIO functional description 454
12.3.1 General-purpose I/O (GPIO) 456
12.3.2 I/O pin alternate function multiplexer and mapping 457
12.3.3 I/O port control registers 457
12.3.4 I/O port state in low-power modes 458
12.3.5 I/O port data registers 458
12.3.6 I/O data bitwise handling 458
12.3.7 GPIO locking mechanism 458
12.3.8 I/O alternate function input/output 459
12.3.9 Input configuration 459
12.3.10 Output configuration 460
12.3.11 Alternate function configuration 460
12.3.12 Analog configuration 461
12.3.13 Using HSE or LSE oscillator pins as GPIOs 461
12.3.14 Using the GPIO pins in the RTC supply domain 462
12.3.15 Using BOOT0 pin as GPIO 462
12.3.16 TrustZone security 462
12.3.17 Privileged and unprivileged modes 463
12.3.18 High-speed low-voltage mode (HSLV) 463
12.3.19 I/O compensation cell 464
12.4 GPIO in low-power modes 464
12.5 GPIO interrupts 464
12.6 GPIO registers 465
12.6.1 GPIO port mode register (GPIOx_MODER) (x = A:E, G, H) 465
12.6.2 GPIO port output type register (GPIOx_OTYPER) (x = A:E, G, H) . . . 466
12.6.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A:E, G, H) 466
12.6.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A:E, G, H) 467
12.6.5 GPIO port input data register (GPIOx_IDR) (x = A:E, G, H) 468
12.6.6 GPIO port output data register (GPIOx_ODR) (x = A:E, G, H) 468
12.6.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A:E, G, H) 469
12.6.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A:E, G, H) . 470
12.6.9 GPIO alternate function low register (GPIOx_AFRL) (x = A:E, G, H) . 471
12.6.10 GPIO alternate function high register (GPIOx_AFRH) (x = A:E, G, H) 471
12.6.11 GPIO port bit reset register (GPIOx_BRR) (x = A:E, G, H) 472
12.6.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A:E, G, H) 473

12.6.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A:E, G, H) . . . 474
12.6.14 GPIO register map . . . 474

13 System configuration controller (SYSCFG) . . . 476
13.1 SYSCFG main features . . . 476
13.2 SYSCFG functional description . . . 476
13.2.1 I/O compensation cell management . . . 476
13.2.2 SYSCFG TrustZone security and privilege . . . 477
13.2.3 GPIO analog switches supply selection . . . 478
13.3 SYSCFG registers . . . 479
13.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR) . . . 479
13.3.2 SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . 480
13.3.3 SYSCFG FPU interrupt mask register (SYSCFG_FPUIMR) . . . 481
13.3.4 SYSCFG CPU nonsecure lock register (SYSCFG_CNSLCKR) . . . 482
13.3.5 SYSCFG CPU secure lock register (SYSCFG_CSLCKR) . . . 483
13.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . 484
13.3.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR) . . . 485
13.3.8 SYSCFG compensation cell value register (SYSCFG_CCVR) . . . 486
13.3.9 SYSCFG compensation cell code register (SYSCFG_CCCR) . . . 487
13.3.10 SYSCFG RSS command register (SYSCFG_RSSCMDR) . . . 488
13.3.11 SYSCFG register map . . . 488

14 Peripherals interconnect matrix . . . 490
14.1 Interconnect matrix introduction . . . 490
14.2 Connection summary . . . 490
14.3 Interconnection details . . . 492
14.3.1 Master to slave interconnection for timers . . . 492
14.3.2 Triggers to ADCs . . . 492
14.3.3 ADC analog watchdogs as triggers to timers . . . 493
14.3.4 Triggers to DAC . . . 494
14.3.5 Triggers to ADF1 . . . 494
14.3.6 Data input to ADF1 . . . 494
14.3.7 Clock sources to timers . . . 495
14.3.8 Triggers to low-power timers . . . 496
14.3.9 Blanking sources to comparators . . . 496

14.3.10 RTC wake-up as inputs to timers . . . 497
14.3.11 USB SOF as trigger to timers . . . 497
14.3.12 Comparators as inputs, trigger or break signals to timers . . . 497
14.3.13 System errors as break signals to timers . . . 498
14.3.14 Timers generating IRTIM signal . . . 499
14.3.15 Triggers for communication peripherals . . . 499
14.3.16 Triggers to GPDMA . . . 500
14.3.17 Internal analog signals to analog peripherals . . . 501
14.3.18 Clock source for the DAC sample and hold mode . . . 501
14.3.19 Internal tamper sources . . . 502
14.3.20 Output from tamper to RTC . . . 502
14.3.21 Encryption keys to AES/SAES . . . 502
14.3.22 I3C IBI acknowledge to timers from I3C1/2 to TIM15/16/17 and LPTIM1/2/3 . . . 503

15 General purpose direct memory access controller (GPDMA) . . . 504

15.1 GPDMA introduction . . . 504
15.2 GPDMA main features . . . 504
15.3 GPDMA implementation . . . 505
15.3.1 GPDMA channels . . . 505
15.3.2 GPDMA in low-power modes . . . 506
15.3.3 GPDMA requests . . . 506
15.3.4 GPDMA block requests . . . 509
15.3.5 GPDMA channels with peripheral early termination . . . 510
15.3.6 GPDMA triggers . . . 510
15.4 GPDMA functional description . . . 512
15.4.1 GPDMA block diagram . . . 512
15.4.2 GPDMA channel state and direct programming without any linked-list 512
15.4.3 GPDMA channel suspend and resume . . . 513
15.4.4 GPDMA channel abort and restart . . . 514
15.4.5 GPDMA linked-list data structure . . . 515
15.4.6 Linked-list item transfer execution . . . 518
15.4.7 GPDMA channel state and linked-list programming in run-to-completion mode . . . 519
15.4.8 GPDMA channel state and linked-list programming in link step mode 522
15.4.9 GPDMA channel state and linked-list programming . . . 529
15.4.10 GPDMA FIFO-based transfers . . . 531

15.4.11 GPDMA transfer request and arbitration 538
15.4.12 GPDMA triggered transfer 542
15.4.13 GPDMA circular buffering with linked-list programming 543
15.4.14 GPDMA transfer in peripheral flow-control mode 545
15.4.15 GPDMA secure/nonsecure channel 546
15.4.16 GPDMA privileged/unprivileged channel 547
15.4.17 GPDMA error management 547
15.5 GPDMA in debug mode 549
15.6 GPDMA in low-power modes 549
15.7 GPDMA interrupts 550
15.8 GPDMA registers 551
15.8.1 GPDMA secure configuration register (GPDMA_SECCFGR) 551
15.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR) 552
15.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR) 552
15.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR) 553
15.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR) . . 554
15.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) 554
15.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR) 555
15.8.8 GPDMA channel x status register (GPDMA_CxSR) 556
15.8.9 GPDMA channel x control register (GPDMA_CxCR) 557
15.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1) 559
15.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2) 563
15.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1) 567
15.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) 568
15.8.14 GPDMA channel x source address register (GPDMA_CxSAR) 571
15.8.15 GPDMA channel x destination address register (GPDMA_CxDAR) . . 572
15.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3) 573
15.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2) 574
15.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR) 575
15.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) 577
15.8.20 GPDMA register map 578

16 Nested vectored interrupt controller (NVIC) 580
16.1 NVIC main features 580

16.2 SysTick calibration value register 580
16.3 Interrupt and exception vectors 581

17 Extended interrupts and event controller (EXTI) 587
17.1 EXTI introduction 587
17.2 EXTI main features 587
17.3 EXTI functional description 587
17.3.1 EXTI block diagram 587
17.3.2 Event features control 589
17.3.3 EXTI configurable event input wake-up 590
17.3.4 EXTI mux selection 590
17.3.5 EXTI functional behavior 591
17.3.6 EXTI event protection 592
17.4 EXTI registers 593
17.4.1 EXTI rising trigger selection register (EXTI_RTSR1) 593
17.4.2 EXTI falling trigger selection register (EXTI_FTSR1) 594
17.4.3 EXTI software interrupt event register (EXTI_SWIER1) 594
17.4.4 EXTI rising edge pending register (EXTI_RPR1) 595
17.4.5 EXTI falling edge pending register (EXTI_FPR1) 596
17.4.6 EXTI security configuration register (EXTI_SECCFGR1) 596
17.4.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) 597
17.4.8 EXTI external interrupt selection register (EXTI_EXTICRx) 597
17.4.9 EXTI lock register (EXTI_LOCKR) 600
17.4.10 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) 600
17.4.11 EXTI CPU wake-up with event mask register (EXTI_EMR1) 601
17.4.12 EXTI register map 601

18 Cyclic redundancy check calculation unit (CRC) 603
18.1 Introduction 603
18.2 CRC main features 603
18.3 CRC functional description 604
18.3.1 CRC block diagram 604
18.3.2 CRC internal signals 604
18.3.3 CRC operation 604
18.4 CRC registers 606
18.4.1 CRC data register (CRC_DR) 606

18.4.2 CRC independent data register (CRC_IDR) 606
18.4.3 CRC control register (CRC_CR) 607
18.4.4 CRC initial value (CRC_INIT) 608
18.4.5 CRC polynomial (CRC_POL) 608
18.4.6 CRC register map 609

19 Extended-SPI interface (OCTOSPI) 610

19.1 OCTOSPI introduction 610
19.2 OCTOSPI main features 610
19.3 OCTOSPI implementation 611
19.4 OCTOSPI functional description 612
19.4.1 OCTOSPI block diagram 612
19.4.2 OCTOSPI pins and internal signals 613
19.4.3 OCTOSPI interface to memory modes 614
19.4.4 OCTOSPI regular-command protocol 614
19.4.5 OCTOSPI regular-command protocol signal interface 618
19.4.6 HyperBus protocol 621
19.4.7 Specific features 625
19.4.8 OCTOSPI operating mode introduction 626
19.4.9 OCTOSPI indirect mode 626
19.4.10 OCTOSPI automatic status-polling mode 628
19.4.11 OCTOSPI memory-mapped mode 629
19.4.12 OCTOSPI configuration introduction 630
19.4.13 OCTOSPI system configuration 630
19.4.14 OCTOSPI device configuration 630
19.4.15 OCTOSPI regular-command mode configuration 633
19.4.16 OCTOSPI HyperBus protocol configuration 635
19.4.17 OCTOSPI error management 636
19.4.18 OCTOSPI BUSY and ABORT 637
19.4.19 OCTOSPI reconfiguration or deactivation 637
19.4.20 NCS behavior 637
19.5 Address alignment and data number 639
19.6 OCTOSPI interrupts 640
19.7 OCTOSPI registers 641
19.7.1 OCTOSPI control register (OCTOSPI_CR) 641
19.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1) 643

19.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2) 645
19.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3) 646
19.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4) 646
19.7.6 OCTOSPI status register (OCTOSPI_SR) 647
19.7.7 OCTOSPI flag clear register (OCTOSPI_FCR) 648
19.7.8 OCTOSPI data length register (OCTOSPI_DLR) 648
19.7.9 OCTOSPI address register (OCTOSPI_AR) 649
19.7.10 OCTOSPI data register (OCTOSPI_DR) 649
19.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR) 650
19.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR) 651
19.7.13 OCTOSPI polling interval register (OCTOSPI_PIR) 651
19.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR) . . 651
19.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR) 654
19.7.16 OCTOSPI instruction register (OCTOSPI_IR) 654
19.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR) 655
19.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR) 655
19.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR) 656
19.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR) . . 658
19.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR) 658
19.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR) 659
19.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR) 659
19.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR) 661
19.7.25 OCTOSPI write instruction register (OCTOSPI_WIR) 662
19.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR) 662
19.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR) 663
19.7.28 OCTOSPI register map 663

20 Secure digital input/output MultiMediaCard interface (SDMMC) . . . 667

20.1 SDMMC main features 667
20.2 SDMMC implementation 667
20.3 SDMMC bus topology 668
20.4 SDMMC operation modes 670
20.5 SDMMC functional description 671
20.5.1 SDMMC block diagram 671

20.5.2 SDMMC pins and internal signals . . . 671
20.5.3 General description . . . 672
20.5.4 SDMMC adapter . . . 674
20.5.5 SDMMC AHB slave interface . . . 696
20.5.6 SDMMC AHB master interface . . . 697
20.5.7 AHB and SDMMC_CK clock relation . . . 700
20.6 Card functional description . . . 700
20.6.1 SD I/O mode . . . 700
20.6.2 CMD12 send timing . . . 708
20.6.3 Sleep (CMD5) . . . 712
20.6.4 Interrupt mode (Wait-IRQ) . . . 713
20.6.5 Boot operation . . . 714
20.6.6 Response R1b handling . . . 717
20.6.7 Reset and card cycle power . . . 718
20.7 Hardware flow control . . . 719
20.8 Ultra-high-speed phase I (UHS-I) voltage switch . . . 719
20.9 SDMMC interrupts . . . 722
20.10 SDMMC registers . . . 724
20.10.1 SDMMC power control register (SDMMC_POWER) . . . 724
20.10.2 SDMMC clock control register (SDMMC_CLKCR) . . . 725
20.10.3 SDMMC argument register (SDMMC_ARGR) . . . 727
20.10.4 SDMMC command register (SDMMC_CMDR) . . . 727
20.10.5 SDMMC command response register (SDMMC_RESPCMDR) . . . 729
20.10.6 SDMMC response x register (SDMMC_RESPxR) . . . 730
20.10.7 SDMMC data timer register (SDMMC_DTIMER) . . . 730
20.10.8 SDMMC data length register (SDMMC_DLENR) . . . 731
20.10.9 SDMMC data control register (SDMMC_DCTRL) . . . 732
20.10.10 SDMMC data counter register (SDMMC_DCNTR) . . . 733
20.10.11 SDMMC status register (SDMMC_STAR) . . . 734
20.10.12 SDMMC interrupt clear register (SDMMC_ICR) . . . 737
20.10.13 SDMMC mask register (SDMMC_MASKR) . . . 739
20.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 742
20.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR) . . . 742
20.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) . . . 743
20.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER) . . . 744
20.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR) . . . 744

20.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR) . . . 745
20.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx) . . . 746
20.10.21 SDMMC register map . . . 746

21 Delay block (DLYB) . . . 749
21.1 Introduction . . . 749
21.2 DLYB main features . . . 749
21.3 DLYB implementation . . . 749
21.4 DLYB functional description . . . 749
21.4.1 DLYB diagram . . . 749
21.4.2 DLYB pins and internal signals . . . 750
21.4.3 General description . . . 750
21.4.4 Delay line length configuration procedure . . . 751
21.4.5 Output clock phase configuration procedure . . . 752
21.5 DLYB registers . . . 752
21.5.1 DLYB control register (DLYB_CR) . . . 752
21.5.2 DLYB configuration register (DLYB_CFGR) . . . 753
21.5.3 DLYB register map . . . 754

22 Analog-to-digital converters (ADC) . . . 755
22.1 Introduction . . . 755
22.2 ADC main features . . . 755
22.3 ADC implementation . . . 757
22.4 ADC functional description . . . 758
22.4.1 ADC block diagram . . . 758
22.4.2 ADC pins and internal signals . . . 759
22.4.3 ADC clocks . . . 762
22.4.4 ADC connectivity . . . 763
22.4.5 Slave AHB interface . . . 765
22.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) . . . 765
22.4.7 Calibration (ADCAL) . . . 765
22.4.8 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 767
22.4.9 Constraints when writing the ADC control bits . . . 768
22.4.10 Channel selection (ADC_SQRy, ADC_JSQR) . . . 769
22.4.11 Channel preselection register (ADC_PCSEL) . . . 770

22.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2) 770
22.4.13 Single conversion mode 772
22.4.14 Continuous conversion mode (CONT = 1) 773
22.4.15 Starting conversions (ADSTART, JADSTART) 773
22.4.16 Timing 774
22.4.17 Stopping an ongoing conversion (ADSTP, JADSTP) 775
22.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) 777
22.4.19 Injected channel management 778
22.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) 779
22.4.21 Queue of context for injected conversions 781
22.4.22 Programmable resolution (RES) - fast conversion mode 789
22.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) 790
22.4.24 End of conversion sequence (EOS, JEOS) 790
22.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers) 791
22.4.26 Data management 793
22.4.27 Managing conversions using the ADF 801
22.4.28 Dynamic low-power features 802
22.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) 806
22.4.30 Oversampler 811
22.4.31 Dual ADC modes 817
22.4.32 Temperature sensor 832
22.4.33 VBAT supply monitoring 834
22.4.34 Monitoring the internal voltage reference 835
22.4.35 Monitoring the DAC output voltage 837
22.4.36 Monitoring the supply voltage 837
22.5 ADC in low-power modes 838
22.6 ADC interrupts 838
22.7 ADC registers (for each ADC) 839
22.7.1 ADC interrupt and status register (ADC_ISR) 839
22.7.2 ADC interrupt enable register (ADC_IER) 842
22.7.3 ADC control register (ADC_CR) 844
22.7.4 ADC configuration register (ADC_CFGR1) 847
22.7.5 ADC configuration register 2 (ADC_CFGR2) 851

22.7.6 ADC sample time register 1 (ADC_SMPR1) 853
22.7.7 ADC sample time register 2 (ADC_SMPR2) 854
22.7.8 ADC channel preselection register (ADC_PCSEL) 855
22.7.9 ADC regular sequence register 1 (ADC_SQR1) 855
22.7.10 ADC regular sequence register 2 (ADC_SQR2) 856
22.7.11 ADC regular sequence register 3 (ADC_SQR3) 857
22.7.12 ADC regular sequence register 4 (ADC_SQR4) 858
22.7.13 ADC regular data register (ADC_DR) 859
22.7.14 ADC injected sequence register (ADC_JSQR) 859
22.7.15 ADC offset y configuration register (ADC_OFCFGRy) 861
22.7.16 ADC offset y register (ADC_OFRy) 862
22.7.17 ADC gain compensation register (ADC_GCOMP) 862
22.7.18 ADC injected channel y data register (ADC_JDRy) 863
22.7.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) 864
22.7.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) 864
22.7.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) . 865
22.7.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) 865
22.7.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) . 866
22.7.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) 866
22.7.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) . 867
22.7.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) 867
22.7.27 ADC calibration factors (ADC_CALFACT) 867
22.7.28 ADC option register (ADC_OR) 869
22.8 ADC common registers 869
22.8.1 ADC common status register (ADCC_CSR) 869
22.8.2 ADC common control register (ADCC_CCR) 871
22.8.3 ADC common regular data register for dual mode (ADCC_CDR) 873
22.8.4 ADC common regular data register for dual mode (ADCC_CDR2) 873
22.9 ADC register map 874

23 Digital-to-analog converter (DAC) 878

23.1 Introduction 878

23.2 DAC main features . . . 878
23.3 DAC implementation . . . 879
23.4 DAC functional description . . . 880
23.4.1 DAC block diagram . . . 880
23.4.2 DAC pins and internal signals . . . 881
23.4.3 DAC clocks . . . 882
23.4.4 DAC channel enable . . . 882
23.4.5 DAC data format . . . 883
23.4.6 DAC conversion . . . 884
23.4.7 DAC output voltage . . . 885
23.4.8 DAC trigger selection . . . 886
23.4.9 DMA requests . . . 886
23.4.10 Noise generation . . . 887
23.4.11 Triangle-wave generation . . . 888
23.4.12 DAC channel modes . . . 889
23.4.13 DAC channel buffer calibration . . . 892
23.4.14 DAC channel conversion modes . . . 893
23.4.15 Dual DAC channel conversion modes (if dual channels are available) . . . 894
23.4.16 DAC autonomous mode . . . 898
23.5 DAC in low-power modes . . . 899
23.6 DAC interrupts . . . 900
23.7 DAC registers . . . 900
23.7.1 DAC control register (DAC_CR) . . . 900
23.7.2 DAC software trigger register (DAC_SWTRGR) . . . 903
23.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) . . . 904
23.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) . . . 905
23.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 905
23.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . 906
23.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . 906
23.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . 907
23.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . 907

23.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . 908
23.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 908
23.7.12 DAC channel1 data output register (DAC_DOR1) . . . 909
23.7.13 DAC channel2 data output register (DAC_DOR2) . . . 909
23.7.14 DAC status register (DAC_SR) . . . 910
23.7.15 DAC calibration control register (DAC_CCR) . . . 911
23.7.16 DAC mode control register (DAC_MCR) . . . 912
23.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 913
23.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) . . . 914
23.7.19 DAC sample and hold time register (DAC_SHHR) . . . 914
23.7.20 DAC sample and hold refresh time register (DAC_SHRR) . . . 915
23.7.21 DAC autonomous mode control register (DAC_AUTOCR) . . . 916
23.7.22 DAC register map . . . 916

24 Voltage reference buffer (VREFBUF) . . . 919

24.1 Introduction . . . 919
24.2 VREFBUF implementation . . . 919
24.3 VREFBUF functional description . . . 919
24.4 VREFBUF trimming . . . 920
24.5 VREFBUF registers . . . 921
24.5.1 VREFBUF control and status register (VREFBUF_CSR) . . . 921
24.5.2 VREFBUF calibration control register (VREFBUF_CCR) . . . 922
24.5.3 VREFBUF register map . . . 922

25 Comparator (COMP) . . . 923

25.1 Introduction . . . 923
25.2 COMP main features . . . 923
25.3 COMP functional description . . . 924
25.3.1 COMP block diagram . . . 924
25.3.2 COMP pins and internal signals . . . 924
25.3.3 Comparator LOCK mechanism . . . 926
25.3.4 Window comparator . . . 926
25.3.5 Hysteresis . . . 927

25.3.6 Comparator output-blanking function . . . 927
25.3.7 COMP power and speed modes . . . 928
25.3.8 Scaler function . . . 928
25.4 COMP in low-power modes . . . 929
25.5 COMP interrupts . . . 929
25.6 COMP registers . . . 930
25.6.1 COMP1 control and status register (COMP1_CSR) . . . 930
25.6.2 COMP2 control and status register (COMP2_CSR) . . . 932
25.6.3 COMP register map . . . 933

26 Operational amplifier (OPAMP) . . . 934
26.1 OPAMP introduction . . . 934
26.2 OPAMP main features . . . 934
26.3 OPAMP functional description . . . 934
26.3.1 OPAMP reset and clocks . . . 934
26.3.2 Initial configuration . . . 935
26.3.3 Signal routing . . . 935
26.3.4 OPAMP modes . . . 936
26.3.5 Calibration . . . 939
26.4 OPAMP low-power modes . . . 941
26.5 OPAMP registers . . . 941
26.5.1 OPAMP1 control/status register (OPAMP1_CSR) . . . 941
26.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1_OTR) . . 943
26.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR) . . . 943
26.5.4 OPAMP2 control/status register (OPAMP2_CRS) . . . 944
26.5.5 OPAMP2 offset trimming register in normal mode (OPAMP2_OTR) . . . 945
26.5.6 OPAMP2 offset trimming register in low-power mode (OPAMP2_LPOTR) . . . 945
26.5.7 OPAMP register map . . . 946

27 Audio digital filter (ADF) . . . 947
27.1 ADF introduction . . . 947
27.2 ADF main features . . . 947
27.3 ADF implementation . . . 948
27.4 ADF functional description . . . 949

27.4.1 ADF block diagram . . . 949
27.4.2 ADF pins and internal signals . . . 949
27.4.3 Serial input interface (SITF) . . . 950
27.4.4 ADC slave interface (ADCITF) . . . 955
27.4.5 Clock generator (CKGEN) . . . 956
27.4.6 Bitstream matrix (BSMX) . . . 958
27.4.7 Digital filter processing (DFLT) . . . 959
27.4.8 Digital filter acquisition modes . . . 968
27.4.9 Start-up sequence examples . . . 976
27.4.10 Sound activity detection (SAD) . . . 977
27.4.11 Data transfer to memory . . . 985
27.4.12 Autonomous mode . . . 988
27.4.13 Register protection . . . 988
27.5 ADF low-power modes . . . 989
27.6 ADF interrupts . . . 989
27.7 ADF application information . . . 991
27.7.1 ADF configuration examples for audio capture . . . 991
27.7.2 Programming examples . . . 992
27.7.3 Connection examples . . . 994
27.7.4 Global frequency response . . . 995
27.7.5 Total ADF gain . . . 996
27.7.6 How to compute SAD thresholds . . . 999
27.8 ADF registers . . . 1003
27.8.1 ADF global control register (ADF_GCR) . . . 1003
27.8.2 ADF clock generator control register (ADF_CKGCR) . . . 1004
27.8.3 ADF trigger input selection register (ADF_TRGISEL) . . . 1006
27.8.4 ADF serial interface control register 0 (ADF_SITF0CR) . . . 1007
27.8.5 ADF bitstream matrix control register 0 (ADF_BSMX0CR) . . . 1008
27.8.6 ADF digital filter control register 0 (ADF_DFLT0CR) . . . 1008
27.8.7 ADF digital filer configuration register 0 (ADF_DFLT0CICR) . . . 1010
27.8.8 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR) . . . 1012
27.8.9 ADF delay control register 0 (ADF_DLY0CR) . . . 1013
27.8.10 ADF DFLT0 interrupt enable register (ADF_DFLT0IER) . . . 1013
27.8.11 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR) . . . 1014
27.8.12 ADF SAD control register (ADF_SADCR) . . . 1015
27.8.13 ADF SAD configuration register (ADF_SADCFGR) . . . 1017
27.8.14 ADF SAD sound level register (ADF_SADSDLVR) . . . 1019

27.8.15 ADF SAD ambient noise level register (ADF_SADANLVR) 1019
27.8.16 ADF digital filter data register 0 (ADF_DFLT0DR) 1020
27.8.17 ADF register map 1020

28 Touch sensing controller (TSC) 1023
28.1 Introduction 1023
28.2 TSC main features 1023
28.3 TSC functional description 1024
28.3.1 TSC block diagram 1024
28.3.2 Surface charge transfer acquisition overview 1024
28.3.3 Reset and clocks 1027
28.3.4 Charge transfer acquisition sequence 1027
28.3.5 Spread spectrum feature 1028
28.3.6 Max count error 1029
28.3.7 Sampling capacitor I/O and channel I/O mode selection 1029
28.3.8 Acquisition mode 1030
28.3.9 I/O hysteresis and analog switch control 1030
28.4 TSC low-power modes 1031
28.5 TSC interrupts 1031
28.6 TSC registers 1031
28.6.1 TSC control register (TSC_CR) 1031
28.6.2 TSC interrupt enable register (TSC_IER) 1034
28.6.3 TSC interrupt clear register (TSC_ICR) 1035
28.6.4 TSC interrupt status register (TSC_ISR) 1035
28.6.5 TSC I/O hysteresis control register (TSC_IOHCR) 1036
28.6.6 TSC I/O analog switch control register (TSC_IOASCR) 1036
28.6.7 TSC I/O sampling control register (TSC_IOSCR) 1037
28.6.8 TSC I/O channel control register (TSC_IOCCR) 1037
28.6.9 TSC I/O group control status register (TSC_IOGCSR) 1038
28.6.10 TSC I/O group x counter register (TSC_IOGxCR) 1038
28.6.11 TSC register map 1039

29 Coupling and chaining bridge (CCB) 1041
29.1 CCB introduction 1041
29.2 CCB main features 1041

29.3 CCB implementation 1042
29.4 CCB functional description 1042
29.4.1 CCB block diagram 1042
29.4.2 CCB internal signals 1043
29.4.3 CCB reset and clocks 1043
29.4.4 CCB coupling and chaining 1043
29.5 CCB coupling and chaining operations 1046
29.5.1 Preliminary steps before any protected operation 1048
29.5.2 ECDSA signature 1048
29.5.3 ECC scalar multiplication 1054
29.5.4 RSA modular exponentiation 1057
29.5.5 CCB error 1059
29.6 CCB interrupts 1059
29.7 CCB registers 1060
29.7.1 CCB control register (CCB_CR) 1060
29.7.2 CCB status register (CCB_SR) 1061
29.7.3 CCB reference tag register (CCB_REFTAGRx) 1063
29.7.4 CCB register map 1064

30 True random number generator (RNG) 1065
30.1 Introduction 1065
30.2 RNG main features 1065
30.3 RNG functional description 1066
30.3.1 RNG block diagram 1066
30.3.2 RNG internal signals 1066
30.3.3 Random number generation 1067
30.3.4 RNG initialization 1070
30.3.5 RNG operation 1071
30.3.6 RNG clocking 1072
30.3.7 Error management 1072
30.3.8 RNG low-power use 1073
30.4 RNG interrupts 1074
30.5 RNG processing time 1074
30.6 RNG entropy source validation 1075
30.6.1 Introduction 1075
30.6.2 Validation conditions 1075

30.7 RNG registers . . . 1076
30.7.1 RNG control register (RNG_CR) . . . 1076
30.7.2 RNG status register (RNG_SR) . . . 1078
30.7.3 RNG data register (RNG_DR) . . . 1079
30.7.4 RNG noise source control register (RNG_NSCR) . . . 1080
30.7.5 RNG health test control register x (RNG_HTCRx) . . . 1080
30.7.6 RNG register map . . . 1081

31 AES hardware accelerator (AES) . . . 1082
31.1 Introduction . . . 1082
31.2 AES main features . . . 1082
31.3 AES implementation . . . 1083
31.4 AES functional description . . . 1083
31.4.1 AES block diagram . . . 1083
31.4.2 AES internal signals . . . 1084
31.4.3 AES reset and clocks . . . 1084
31.4.4 AES symmetric cipher implementation . . . 1084
31.4.5 AES encryption or decryption typical usage . . . 1085
31.4.6 AES authenticated encryption, decryption, and cipher-based message authentication . . . 1087
31.4.7 AES ciphertext stealing and data padding . . . 1087
31.4.8 AES suspend and resume operations . . . 1088
31.4.9 AES basic chaining modes (ECB, CBC) . . . 1089
31.4.10 AES counter (CTR) mode . . . 1092
31.4.11 AES Galois/counter mode (GCM) . . . 1095
31.4.12 AES Galois message authentication code (GMAC) . . . 1099
31.4.13 AES counter with CBC-MAC (CCM) . . . 1100
31.4.14 AES key sharing with secure AES co-processor . . . 1105
31.4.15 AES data registers and data swapping . . . 1106
31.4.16 AES key registers . . . 1108
31.4.17 AES initialization vector registers . . . 1108
31.4.18 AES error management . . . 1109
31.5 AES interrupts . . . 1110
31.6 AES DMA requests . . . 1110
31.7 AES processing latency . . . 1111
31.8 AES registers . . . 1113

31.8.1 AES control register (AES_CR) 1113
31.8.2 AES status register (AES_SR) 1115
31.8.3 AES data input register (AES_DINR) 1116
31.8.4 AES data output register (AES_DOUTR) 1117
31.8.5 AES key register 0 (AES_KEYR0) 1117
31.8.6 AES key register 1 (AES_KEYR1) 1118
31.8.7 AES key register 2 (AES_KEYR2) 1118
31.8.8 AES key register 3 (AES_KEYR3) 1118
31.8.9 AES initialization vector register 0 (AES_IVR0) 1119
31.8.10 AES initialization vector register 1 (AES_IVR1) 1119
31.8.11 AES initialization vector register 2 (AES_IVR2) 1119
31.8.12 AES initialization vector register 3 (AES_IVR3) 1120
31.8.13 AES key register 4 (AES_KEYR4) 1120
31.8.14 AES key register 5 (AES_KEYR5) 1120
31.8.15 AES key register 6 (AES_KEYR6) 1121
31.8.16 AES key register 7 (AES_KEYR7) 1121
31.8.17 AES suspend registers (AES_SUSPRx) 1121
31.8.18 AES interrupt enable register (AES_IER) 1122
31.8.19 AES interrupt status register (AES_ISR) 1123
31.8.20 AES interrupt clear register (AES_ICR) 1124
31.8.21 AES register map 1124

32 Secure AES coprocessor (SAES) 1127

32.1 Introduction 1127
32.2 SAES main features 1127
32.3 SAES implementation 1128
32.4 SAES functional description 1128
32.4.1 SAES block diagram 1128
32.4.2 SAES internal signals 1129
32.4.3 SAES reset and clocks 1130
32.4.4 SAES symmetric cipher implementation 1130
32.4.5 SAES encryption or decryption typical usage 1131
32.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication 1133
32.4.7 SAES ciphertext stealing and data padding 1134
32.4.8 SAES suspend and resume operations 1134
32.4.9 SAES basic chaining modes (ECB, CBC) 1135

32.4.10 SAES counter (CTR) mode 1139
32.4.11 SAES Galois/counter mode (GCM) 1141
32.4.12 SAES Galois message authentication code (GMAC) 1145
32.4.13 SAES counter with CBC-MAC (CCM) 1147
32.4.14 SAES operation with wrapped keys 1152
32.4.15 SAES operation with shared keys 1156
32.4.16 SAES data registers and data swapping 1157
32.4.17 SAES key registers 1160
32.4.18 SAES initialization vector registers 1161
32.4.19 SAES error management 1162
32.5 SAES interrupts 1164
32.6 SAES DMA requests 1164
32.7 SAES processing latency 1165
32.8 SAES registers 1167
32.8.1 SAES control register (SAES_CR) 1167
32.8.2 SAES status register (SAES_SR) 1170
32.8.3 SAES data input register (SAES_DINR) 1171
32.8.4 SAES data output register (SAES_DOUTR) 1172
32.8.5 SAES key register 0 (SAES_KEYR0) 1172
32.8.6 SAES key register 1 (SAES_KEYR1) 1173
32.8.7 SAES key register 2 (SAES_KEYR2) 1173
32.8.8 SAES key register 3 (SAES_KEYR3) 1173
32.8.9 SAES initialization vector register 0 (SAES_IVR0) 1174
32.8.10 SAES initialization vector register 1 (SAES_IVR1) 1174
32.8.11 SAES initialization vector register 2 (SAES_IVR2) 1174
32.8.12 SAES initialization vector register 3 (SAES_IVR3) 1175
32.8.13 SAES key register 4 (SAES_KEYR4) 1175
32.8.14 SAES key register 5 (SAES_KEYR5) 1175
32.8.15 SAES key register 6 (SAES_KEYR6) 1176
32.8.16 SAES key register 7 (SAES_KEYR7) 1176
32.8.17 SAES suspend registers (SAES_SUSPRx) 1176
32.8.18 SAES interrupt enable register (SAES_IER) 1177
32.8.19 SAES interrupt status register (SAES_ISR) 1178
32.8.20 SAES interrupt clear register (SAES_ICR) 1179
32.8.21 SAES register map 1180

33 Hash processor (HASH) 1182

33.1 Introduction 1182
33.2 HASH main features 1182
33.3 HASH implementation 1183
33.4 HASH functional description 1183
33.4.1 HASH block diagram 1183
33.4.2 HASH internal signals 1183
33.4.3 About secure hash algorithms 1184
33.4.4 Message data feeding 1184
33.4.5 Message digest computing 1185
33.4.6 Message padding 1187
33.4.7 HMAC operation 1188
33.4.8 HASH suspend/resume operations 1190
33.4.9 HASH DMA interface 1192
33.4.10 HASH error management 1193
33.4.11 HASH processing time 1193
33.5 HASH interrupts 1194
33.6 HASH registers 1194
33.6.1 HASH control register (HASH_CR) 1194
33.6.2 HASH data input register (HASH_DIN) 1196
33.6.3 HASH start register (HASH_STR) 1197
33.6.4 HASH digest registers 1198
33.6.5 HASH interrupt enable register (HASH_IMR) 1200
33.6.6 HASH status register (HASH_SR) 1200
33.6.7 HASH context swap registers 1201
33.6.8 HASH register map 1202

34 Public key accelerator (PKA) 1204
34.1 PKA introduction 1204
34.2 PKA main features 1204
34.3 PKA implementation 1204
34.4 PKA functional description 1205
34.4.1 PKA block diagram 1205
34.4.2 PKA internal signals 1205
34.4.3 PKA reset and clocks 1206
34.4.4 PKA public key acceleration 1206
34.4.5 Typical applications for PKA 1208

34.4.6 PKA procedure to perform an operation 1210
34.4.7 Private keys protection using CCB 1211
34.4.8 PKA error management 1215
34.5 PKA operating modes 1215
34.5.1 Introduction 1215
34.5.2 Montgomery parameter computation 1217
34.5.3 Modular addition 1217
34.5.4 Modular subtraction 1217
34.5.5 Modular and Montgomery multiplication 1218
34.5.6 Modular exponentiation 1219
34.5.7 Modular inversion 1220
34.5.8 Modular reduction 1221
34.5.9 Arithmetic addition 1221
34.5.10 Arithmetic subtraction 1221
34.5.11 Arithmetic multiplication 1222
34.5.12 Arithmetic comparison 1222
34.5.13 RSA CRT exponentiation 1223
34.5.14 Point on elliptic curve Fp check 1223
34.5.15 ECC Fp scalar multiplication 1224
34.5.16 ECDSA sign 1225
34.5.17 ECDSA verification 1227
34.5.18 ECC complete addition 1228
34.5.19 ECC double base ladder 1228
34.5.20 ECC projective to affine 1229
34.6 Example of configurations and processing times 1230
34.6.1 Supported elliptic curves 1230
34.6.2 Computation times 1232
34.7 PKA interrupts 1234
34.8 PKA registers 1235
34.8.1 PKA control register (PKA_CR) 1235
34.8.2 PKA status register (PKA_SR) 1237
34.8.3 PKA clear flag register (PKA_CLRFR) 1239
34.8.4 PKA RAM 1240
34.8.5 PKA register map 1241

35 Advanced-control timers (TIM1) 1242

35.1 TIM1 introduction 1242

35.2 TIM1 main features 1243
35.3 TIM1 functional description 1244
35.3.1 Block diagram 1244
35.3.2 TIM1 pins and internal signals 1245
35.3.3 Time-base unit 1249
35.3.4 Counter modes 1251
35.3.5 Repetition counter 1263
35.3.6 External trigger input 1264
35.3.7 Clock selection 1265
35.3.8 Capture/compare channels 1269
35.3.9 Input capture mode 1272
35.3.10 PWM input mode 1273
35.3.11 Forced output mode 1274
35.3.12 Output compare mode 1274
35.3.13 PWM mode 1276
35.3.14 Asymmetric PWM mode 1284
35.3.15 Combined PWM mode 1285
35.3.16 Combined 3-phase PWM mode 1286
35.3.17 Complementary outputs and dead-time insertion 1287
35.3.18 Using the break function 1290
35.3.19 Bidirectional break inputs 1296
35.3.20 Clearing the tim_ocxref signal on an external event 1297
35.3.21 6-step PWM generation 1299
35.3.22 One-pulse mode 1300
35.3.23 Retriggerable One-pulse mode 1302
35.3.24 Pulse on compare mode 1303
35.3.25 Encoder interface mode 1305
35.3.26 Direction bit output 1322
35.3.27 UIF bit remapping 1323
35.3.28 Timer input XOR function 1323
35.3.29 Interfacing with Hall sensors 1323
35.3.30 Timer synchronization 1325
35.3.31 ADC triggers 1330
35.3.32 ADC synchronization 1330
35.3.33 DMA burst mode 1331
35.3.34 TIM1 DMA requests 1332
35.3.35 Debug mode 1332

35.4 TIM1 low-power modes 1333
35.5 TIM1 interrupts 1333
35.6 TIM1 registers 1334
35.6.1 TIM1 control register 1 (TIM1_CR1) 1334
35.6.2 TIM1 control register 2 (TIM1_CR2) 1335
35.6.3 TIM1 slave mode control register (TIM1_SMCR) 1339
35.6.4 TIM1 DMA/interrupt enable register (TIM1_DIER) 1343
35.6.5 TIM1 status register (TIM1_SR) 1344
35.6.6 TIM1 event generation register (TIM1_EGR) 1347
35.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1) 1348
35.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) 1350
35.6.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2) 1353
35.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) 1354
35.6.11 TIM1 capture/compare enable register (TIM1_CCER) 1357
35.6.12 TIM1 counter (TIM1_CNT) 1361
35.6.13 TIM1 prescaler (TIM1_PSC) 1361
35.6.14 TIM1 autoreload register (TIM1_ARR) 1362
35.6.15 TIM1 repetition counter register (TIM1_RCR) 1362
35.6.16 TIM1 capture/compare register 1 (TIM1_CCR1) 1363
35.6.17 TIM1 capture/compare register 2 (TIM1_CCR2) 1363
35.6.18 TIM1 capture/compare register 3 (TIM1_CCR3) 1364
35.6.19 TIM1 capture/compare register 4 (TIM1_CCR4) 1365
35.6.20 TIM1 break and dead-time register (TIM1_BDTR) 1366
35.6.21 TIM1 capture/compare register 5 (TIM1_CCR5) 1370
35.6.22 TIM1 capture/compare register 6 (TIM1_CCR6) 1371
35.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3) 1372
35.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2) 1373
35.6.25 TIM1 timer encoder control register (TIM1_ECR) 1374
35.6.26 TIM1 timer input selection register (TIM1_TISEL) 1375
35.6.27 TIM1 alternate function option register 1 (TIM1_AF1) 1376
35.6.28 TIM1 alternate function register 2 (TIM1_AF2) 1379
35.6.29 TIM1 DMA control register (TIM1_DCR) 1381
35.6.30 TIM1 DMA address for full transfer (TIM1_DMAR) 1383
35.6.31 TIM1 register map 1383

36 General-purpose timers (TIM2/TIM3/TIM4) 1386

36.1 TIM2/TIM3/TIM4 introduction 1386

36.2 TIM2/TIM3/TIM4 main features 1386

36.3 TIM2/TIM3/TIM4 implementation 1387

36.4 TIM2/TIM3/TIM4 functional description 1388

36.4.1 Block diagram 1388

36.4.2 TIM2/TIM3/TIM4 pins and internal signals 1389

36.4.3 Time-base unit 1392

36.4.4 Counter modes 1394

36.4.5 Clock selection 1406

36.4.6 Capture/compare channels 1410

36.4.7 Input capture mode 1412

36.4.8 PWM input mode 1413

36.4.9 Forced output mode 1414

36.4.10 Output compare mode 1414

36.4.11 PWM mode 1416

36.4.12 Asymmetric PWM mode 1424

36.4.13 Combined PWM mode 1425

36.4.14 Clearing the tim_ocxref signal on an external event 1426

36.4.15 One-pulse mode 1428

36.4.16 Retriggerable one-pulse mode 1429

36.4.17 Pulse on compare mode 1430

36.4.18 Encoder interface mode 1432

36.4.19 Direction bit output 1450

36.4.20 UIF bit remapping 1451

36.4.21 Timer input XOR function 1451

36.4.22 Timers and external trigger synchronization 1451

36.4.23 Timer synchronization 1455

36.4.24 ADC triggers 1460

36.4.25 ADC synchronization 1462

36.4.26 DMA burst mode 1462

36.4.27 TIM2/TIM3/TIM4 DMA requests 1463

36.4.28 Debug mode 1464

36.4.29 TIM2/TIM3/TIM4 low-power modes 1464

36.4.30 TIM2/TIM3/TIM4 interrupts 1465

36.5 TIM2/TIM3/TIM4 registers 1466

36.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 4) 1466
36.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 4) 1467
36.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 4) 1469
36.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 4) 1473
36.5.5 TIMx status register (TIMx_SR)(x = 2 to 4) 1474
36.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 4) 1476
36.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 4) . 1477
36.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 4) 1479
36.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 4) . 1481
36.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 4) 1482
36.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 4) . . . 1485
36.5.12 TIMx counter (TIMx_CNT)(x = 2 to 4) 1487
36.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 4) 1487
36.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 4) 1488
36.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 4) 1488
36.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 4) 1489
36.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 4) 1490
36.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 4) 1491
36.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 4) 1492
36.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 4) 1493
36.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 4) 1494
36.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 4) 1495
36.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 4) 1496
36.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 4) 1497
36.5.25 TIMx register map 1498

37 Basic timers (TIM6/TIM7) 1501

37.1 TIM6/TIM7 introduction 1501
37.2 TIM6/TIM7 main features 1501
37.3 TIM6/TIM7 functional description 1502
37.3.1 TIM6/TIM7 block diagram 1502
37.3.2 TIM6/TIM7 internal signals 1502
37.3.3 TIM6/TIM7 clocks 1503
37.3.4 Time-base unit 1503
37.3.5 Counting mode 1505

37.3.6 UIF bit remapping . . . 1512
37.3.7 ADC triggers . . . 1513
37.3.8 ADC synchronization . . . 1513
37.3.9 TIM6/TIM7 DMA requests . . . 1514
37.3.10 Debug mode . . . 1514
37.3.11 TIM6/TIM7 low-power modes . . . 1514
37.3.12 TIM6/TIM7 interrupts . . . 1514
37.4 TIM6/TIM7 registers . . . 1515
37.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) . . . 1515
37.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) . . . 1517
37.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) . . . 1517
37.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) . . . 1519
37.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) . . . 1519
37.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) . . . 1519
37.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) . . . 1520
37.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7) . . . 1520
37.4.9 TIMx register map . . . 1521

38 General purpose timers (TIM15/TIM16/TIM17) . . . 1522
38.1 TIM15/TIM16/TIM17 introduction . . . 1522
38.2 TIM15 main features . . . 1522
38.3 TIM16/TIM17 main features . . . 1523
38.4 TIM15/TIM16/TIM17 functional description . . . 1524
38.4.1 Block diagram . . . 1524
38.4.2 TIM15/TIM16/TIM17 pins and internal signals . . . 1525
38.4.3 Time-base unit . . . 1528
38.4.4 Counter modes . . . 1530
38.4.5 Repetition counter . . . 1534
38.4.6 Clock selection . . . 1535
38.4.7 Capture/compare channels . . . 1537
38.4.8 Input capture mode . . . 1539
38.4.9 PWM input mode (only for TIM15) . . . 1541
38.4.10 Forced output mode . . . 1542
38.4.11 Output compare mode . . . 1542
38.4.12 PWM mode . . . 1544
38.4.13 Combined PWM mode (TIM15 only) . . . 1549
38.4.14 Complementary outputs and dead-time insertion . . . 1550

38.4.15 Using the break function 1553
38.4.16 Bidirectional break input 1557
38.4.17 Clearing the tim_ocxref signal on an external event 1558
38.4.18 6-step PWM generation 1559
38.4.19 One-pulse mode 1561
38.4.20 Retriggerable one pulse mode (TIM15 only) 1562
38.4.21 UIF bit remapping 1563
38.4.22 Timer input XOR function (TIM15 only) 1563
38.4.23 External trigger synchronization (TIM15 only) 1563
38.4.24 Slave mode – combined reset + trigger mode (TIM15 only) 1566
38.4.25 Slave mode – combined reset + gated mode (TIM15 only) 1566
38.4.26 Timer synchronization (TIM15 only) 1567
38.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 1567
38.4.28 ADC triggers (TIM15 only) 1567
38.4.29 ADC synchronization (TIM15 only) 1568
38.4.30 DMA burst mode 1568
38.4.31 TIM15/TIM16/TIM17 DMA requests 1569
38.4.32 Debug mode 1570
38.5 TIM15/TIM16/TIM17 low-power modes 1570
38.6 TIM15/TIM16/TIM17 interrupts 1570
38.7 TIM15 registers 1572
38.7.1 TIM15 control register 1 (TIM15_CR1) 1572
38.7.2 TIM15 control register 2 (TIM15_CR2) 1573
38.7.3 TIM15 slave mode control register (TIM15_SMCR) 1576
38.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) 1578
38.7.5 TIM15 status register (TIM15_SR) 1579
38.7.6 TIM15 event generation register (TIM15_EGR) 1581
38.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) 1582
38.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 1583
38.7.9 TIM15 capture/compare enable register (TIM15_CCER) 1586
38.7.10 TIM15 counter (TIM15_CNT) 1589
38.7.11 TIM15 prescaler (TIM15_PSC) 1589
38.7.12 TIM15 autoreload register (TIM15_ARR) 1590
38.7.13 TIM15 repetition counter register (TIM15_RCR) 1590
38.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) 1591
38.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) 1592

38.7.16 TIM15 break and dead-time register (TIM15_BDTR) 1592
38.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) 1595
38.7.18 TIM15 input selection register (TIM15_TISEL) 1596
38.7.19 TIM15 alternate function register 1 (TIM15_AF1) 1597
38.7.20 TIM15 alternate function register 2 (TIM15_AF2) 1599
38.7.21 TIM15 DMA control register (TIM15_DCR) 1600
38.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) 1601
38.7.23 TIM15 register map 1601
38.8 TIM16/TIM17 registers 1604
38.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) 1604
38.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) 1605
38.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 1606
38.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) 1607
38.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) 1608
38.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) 1609
38.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) 1610
38.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . 1612
38.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) 1615
38.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) 1615
38.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) 1616
38.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) 1616
38.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) 1617
38.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) 1618
38.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) 1621
38.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) 1622
38.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) 1622
38.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) 1625
38.8.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17) 1625
38.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) 1626
38.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) 1627
38.8.22 TIM16/TIM17 register map 1628

39 Low-power timer (LPTIM) 1630
39.1 Introduction 1630
39.2 LPTIM main features 1630

39.3 LPTIM implementation 1631
39.4 LPTIM functional description 1632
39.4.1 LPTIM block diagram 1632
39.4.2 LPTIM pins and internal signals 1633
39.4.3 LPTIM input and trigger mapping 1635
39.4.4 LPTIM reset and clocks 1636
39.4.5 Glitch filter 1637
39.4.6 Prescaler 1638
39.4.7 Trigger multiplexer 1638
39.4.8 Operating mode 1639
39.4.9 Timeout function 1641
39.4.10 Waveform generation 1641
39.4.11 Register update 1642
39.4.12 Counter mode 1643
39.4.13 Timer enable 1643
39.4.14 Timer counter reset 1644
39.4.15 Encoder mode 1644
39.4.16 Repetition counter 1646
39.4.17 Capture/compare channels 1647
39.4.18 Input capture mode 1648
39.4.19 PWM mode 1650
39.4.20 Autonomous mode 1652
39.4.21 DMA requests 1653
39.4.22 Debug mode 1654
39.5 LPTIM low-power modes 1654
39.6 LPTIM interrupts 1654
39.7 LPTIM registers 1655
39.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR) 1656
39.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) 1657
39.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) 1659
39.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR) 1661
39.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) 1662
39.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) 1663
39.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER) 1664

39.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) 1666
39.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) 1667
39.7.10 LPTIM configuration register (LPTIM_CFGR) 1669
39.7.11 LPTIM control register (LPTIM_CR) 1672
39.7.12 LPTIM compare register 1 (LPTIM_CCR1) 1673
39.7.13 LPTIM autoreload register (LPTIM_ARR) 1674
39.7.14 LPTIM counter register (LPTIM_CNT) 1674
39.7.15 LPTIM configuration register 2 (LPTIM_CFGR2) 1675
39.7.16 LPTIM repetition register (LPTIM_RCR) 1676
39.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) 1676
39.7.18 LPTIM compare register 2 (LPTIM_CCR2) 1679
39.7.19 LPTIM register map 1679

40 Infrared interface (IRTIM) 1682

41 Independent watchdog (IWDG) 1683

41.1 Introduction 1683
41.2 IWDG main features 1683
41.3 IWDG implementation 1684
41.4 IWDG functional description 1684
41.4.1 IWDG block diagram 1684
41.4.2 IWDG internal signals 1685
41.4.3 Software and hardware watchdog modes 1686
41.4.4 Window option 1687
41.4.5 Debug 1690
41.4.6 Register access protection 1690
41.5 IWDG low power modes 1690
41.6 IWDG interrupts 1691
41.7 IWDG registers 1692
41.7.1 IWDG key register (IWDG_KR) 1692
41.7.2 IWDG prescaler register (IWDG_PR) 1693
41.7.3 IWDG reload register (IWDG_RLR) 1694
41.7.4 IWDG status register (IWDG_SR) 1694
41.7.5 IWDG window register (IWDG_WINR) 1695
41.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1696

41.7.7 IWDG register map 1697

42 System window watchdog (WWDG) 1698

42.1 Introduction 1698
42.2 WWDG main features 1698
42.3 WWDG implementation 1698
42.4 WWDG functional description 1699
42.4.1 WWDG block diagram 1699
42.4.2 WWDG internal signals 1699
42.4.3 Enabling the watchdog 1700
42.4.4 Controlling the down-counter 1700
42.4.5 How to program the watchdog timeout 1700
42.4.6 Debug mode 1701
42.5 WWDG interrupts 1702
42.6 WWDG registers 1702
42.6.1 WWDG control register (WWDG_CR) 1702
42.6.2 WWDG configuration register (WWDG_CFR) 1703
42.6.3 WWDG status register (WWDG_SR) 1704
42.6.4 WWDG register map 1704

43 Real-time clock (RTC) 1705

43.1 Introduction 1705
43.2 RTC main features 1705
43.3 RTC functional description 1706
43.3.1 RTC block diagram 1706
43.3.2 RTC pins and internal signals 1708
43.3.3 GPIOs controlled by the RTC 1709
43.3.4 RTC secure protection modes 1709
43.3.5 RTC privilege protection modes 1711
43.3.6 Clock and prescalers 1712
43.3.7 Real-time clock and calendar 1713
43.3.8 Calendar ultra-low power mode 1714
43.3.9 Programmable alarms 1714
43.3.10 Periodic auto-wake-up 1714
43.3.11 RTC initialization and configuration 1715
43.3.12 Reading the calendar 1718

43.3.13 Resetting the RTC . . . 1719
43.3.14 RTC synchronization . . . 1719
43.3.15 RTC reference clock detection . . . 1720
43.3.16 RTC smooth digital calibration . . . 1721
43.3.17 Timestamp function . . . 1723
43.3.18 Calibration clock output . . . 1724
43.3.19 Tamper and alarm output . . . 1724
43.4 RTC low-power modes . . . 1725
43.5 RTC interrupts . . . 1725
43.6 RTC registers . . . 1727
43.6.1 RTC time register (RTC_TR) . . . 1727
43.6.2 RTC date register (RTC_DR) . . . 1728
43.6.3 RTC subsecond register (RTC_SSR) . . . 1729
43.6.4 RTC initialization control and status register (RTC_ICSR) . . . 1730
43.6.5 RTC prescaler register (RTC_PRER) . . . 1732
43.6.6 RTC wake-up timer register (RTC_WUTR) . . . 1733
43.6.7 RTC control register (RTC_CR) . . . 1734
43.6.8 RTC privilege mode control register (RTC_PRIVCFGR) . . . 1738
43.6.9 RTC secure configuration register (RTC_SECCFGR) . . . 1739
43.6.10 RTC write protection register (RTC_WPR) . . . 1740
43.6.11 RTC calibration register (RTC_CALR) . . . 1741
43.6.12 RTC shift control register (RTC_SHIFTR) . . . 1742
43.6.13 RTC timestamp time register (RTC_TSTR) . . . 1743
43.6.14 RTC timestamp date register (RTC_TSDR) . . . 1744
43.6.15 RTC timestamp subsecond register (RTC_TSSSR) . . . 1745
43.6.16 RTC alarm A register (RTC_ALRMAR) . . . 1745
43.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) . . . 1747
43.6.18 RTC alarm B register (RTC_ALRMBR) . . . 1748
43.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) . . . 1749
43.6.20 RTC status register (RTC_SR) . . . 1750
43.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) . . . 1751
43.6.22 RTC secure masked interrupt status register (RTC_SMISR) . . . 1752
43.6.23 RTC status clear register (RTC_SCR) . . . 1754
43.6.24 RTC timestamp on tamper control register (RTC_TAMPTSCR) . . . 1755
43.6.25 RTC timestamp status register (RTC_TSIDR) . . . 1756
43.6.26 RTC alarm A binary mode register (RTC_ALRABINR) . . . 1757
43.6.27 RTC alarm B binary mode register (RTC_ALRBBINR) . . . 1757

43.6.28 RTC register map 1759

44 Tamper and backup registers (TAMP) 1762

44.1 Introduction 1762

44.2 TAMP main features 1762

44.3 TAMP functional description 1763

44.3.1 TAMP block diagram 1763

44.3.2 TAMP pins and internal signals 1764

44.3.3 GPIOs controlled by the RTC and TAMP 1766

44.3.4 TAMP register write protection 1768

44.3.5 TAMP secure protection modes 1768

44.3.6 Backup registers protection zones 1768

44.3.7 TAMP privilege protection modes 1769

44.3.8 Boot hardware key (BHK) 1769

44.3.9 Tamper detection 1769

44.3.10 TAMP backup registers and other device secrets erase 1770

44.3.11 Tamper detection - confirmed mode 1770

44.3.12 Tamper detection configuration and initialization 1772

44.4 TAMP in low-power modes 1775

44.5 TAMP interrupts 1775

44.6 TAMP registers 1776

44.6.1 TAMP control register 1 (TAMP_CR1) 1776

44.6.2 TAMP control register 2 (TAMP_CR2) 1777

44.6.3 TAMP control register 3 (TAMP_CR3) 1778

44.6.4 TAMP filter control register (TAMP_FLTCR) 1779

44.6.5 TAMP secure configuration register (TAMP_SECCFGR) 1780

44.6.6 TAMP privileged configuration register (TAMP_PRIVCFGR) 1782

44.6.7 TAMP interrupt enable register (TAMP_IER) 1783

44.6.8 TAMP status register (TAMP_SR) 1784

44.6.9 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . 1785

44.6.10 TAMP secure masked interrupt status register (TAMP_SMISR) 1786

44.6.11 TAMP status clear register (TAMP_SCR) 1787

44.6.12 TAMP monotonic counter 1 register (TAMP_COUNT1R) 1788

44.6.13 TAMP resource protection register (TAMP_RPCFGR) 1788

44.6.14 TAMP backup x register (TAMP_BKPxR) 1789

44.6.15 TAMP register map 1789

45 Inter-integrated circuit interface (I2C) 1792

45.1 Introduction 1792

45.2 I2C main features 1792

45.3 I2C implementation 1793

45.4 I2C functional description 1793

45.4.1 I2C block diagram 1794

45.4.2 I2C pins and internal signals 1794

45.4.3 I2C clock requirements 1795

45.4.4 I2C mode selection 1796

45.4.5 I2C initialization 1797

45.4.6 I2C reset 1801

45.4.7 I2C data transfer 1801

45.4.8 I2C target mode 1803

45.4.9 I2C controller mode 1812

45.4.10 I2C_TIMINGR register configuration examples 1823

45.4.11 SMBus specific features 1825

45.4.12 SMBus initialization 1828

45.4.13 SMBus I2C_TIMEOUTR register configuration examples 1830

45.4.14 SMBus target mode 1830

45.4.15 SMBus controller mode 1834

45.4.16 Autonomous mode 1837

45.4.17 Error conditions 1839

45.5 I2C in low-power modes 1840

45.6 I2C interrupts 1841

45.7 I2C DMA requests 1841

45.7.1 Transmission using DMA 1841

45.7.2 Reception using DMA 1842

45.7.3 Controller event control using DMA 1842

45.8 I2C debug modes 1843

45.9 I2C registers 1843

45.9.1 I2C control register 1 (I2C_CR1) 1843

45.9.2 I2C control register 2 (I2C_CR2) 1846

45.9.3 I2C own address 1 register (I2C_OAR1) 1848

45.9.4 I2C own address 2 register (I2C_OAR2) 1848

45.9.5 I2C timing register (I2C_TIMINGR) 1849

45.9.6 I2C timeout register (I2C_TIMEOUTR) 1850

45.9.7 I2C interrupt and status register (I2C_ISR) . . . 1851
45.9.8 I2C interrupt clear register (I2C_ICR) . . . 1854
45.9.9 I2C PEC register (I2C_PECR) . . . 1855
45.9.10 I2C receive data register (I2C_RXDR) . . . 1855
45.9.11 I2C transmit data register (I2C_TXDR) . . . 1856
45.9.12 I2C autonomous mode control register (I2C_AUTOCR) . . . 1856
45.9.13 I2C register map . . . 1858

46 Improved inter-integrated circuit (I3C) . . . 1860

46.1 Introduction . . . 1860
46.2 I3C main features . . . 1860
46.3 I3C implementation . . . 1862
46.3.1 I3C instantiation . . . 1862
46.3.2 I3C wake-up from low-power mode(s) . . . 1862
46.3.3 I3C autonomous mode with DMA in Stop mode(s) . . . 1862
46.3.4 I3C FIFOs . . . 1862
46.3.5 I3C triggers . . . 1863
46.3.6 I3C interrupt(s) . . . 1863
46.3.7 I3C MIPI® support . . . 1864
46.4 I3C block diagram . . . 1865
46.5 I3C pins and internal signals . . . 1865
46.6 I3C reset and clocks . . . 1866
46.6.1 I3C reset . . . 1866
46.6.2 I3C clocks and requirements . . . 1866
46.7 I3C peripheral state and programming . . . 1868
46.7.1 I3C peripheral state . . . 1868
46.7.2 I3C controller state and programming sequence . . . 1869
46.7.3 I3C target state and programming sequence . . . 1873
46.8 I3C registers and programming . . . 1877
46.8.1 I3C register set, as controller/target . . . 1877
46.8.2 I3C registers and fields use versus peripheral state, as controller . . . 1878
46.8.3 I3C registers and fields usage versus peripheral state, as target . . . 1881
46.9 I3C bus transfers and programming . . . 1883
46.9.1 I3C command set (CCCs), as controller/target . . . 1883
46.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller . . . 1887

46.9.3 I3C broadcast ENTDAA CCC transfer, as controller 1889
46.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller 1889
46.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target 1891
46.9.6 I3C broadcast ENTDAA CCC transfer, as target 1893
46.9.7 I3C broadcast DEFTGTS CCC transfer, as target 1894
46.9.8 I3C broadcast DEFGRPA CCC transfer, as target 1895
46.9.9 I3C direct GETSTATUS CCC response, as target 1896
46.9.10 I3C private read/write transfer, as controller 1897
46.9.11 I3C private read/write transfer, as target 1898
46.9.12 Legacy I2C read/write transfer, as controller 1899
46.9.13 I3C IBI transfer, as controller/target 1900
46.9.14 I3C hot-join request transfer, as controller/target 1902
46.9.15 I3C controller-role request transfer, as controller/target 1903
46.10 I3C FIFOs management, as controller 1904
46.10.1 C-FIFO management, as controller 1904
46.10.2 TX-FIFO management, as controller 1905
46.10.3 RX-FIFO management, as controller 1908
46.10.4 S-FIFO management, as controller 1910
46.11 I3C FIFOs management, as target 1912
46.11.1 RX-FIFO management, as target 1912
46.11.2 TX-FIFO management, as target 1913
46.12 I3C error management 1916
46.12.1 Controller error management 1916
46.12.2 Target error management 1918
46.13 I3C wake-up from low-power mode(s) 1919
46.13.1 Wake-up from Stop 1919
46.13.2 52.13.1 I3C wake-up from Standby 1922
46.14 I3C in low-power modes 1922
46.15 I3C interrupts 1923
46.16 I3C registers 1924
46.16.1 I3C message control register (I3C_CR) 1924
46.16.2 I3C message control register [alternate] (I3C_CR) 1926
46.16.3 I3C configuration register (I3C_CFGR) 1928
46.16.4 I3C receive data byte register (I3C_RDR) 1933
46.16.5 I3C receive data word register (I3C_RDWR) 1934

46.16.6 I3C transmit data byte register (I3C_TDR) 1935
46.16.7 I3C transmit data word register (I3C_TDWR) 1936
46.16.8 I3C IBI payload data register (I3C_IBIDR) 1938
46.16.9 I3C target transmit configuration register (I3C_TGTTDR) 1939
46.16.10 I3C status register (I3C_SR) 1940
46.16.11 I3C status error register (I3C_SER) 1941
46.16.12 I3C received message register (I3C_RMR) 1943
46.16.13 I3C event register (I3C_EVR) 1944
46.16.14 I3C interrupt enable register (I3C_IER) 1948
46.16.15 I3C clear event register (I3C_CEVR) 1950
46.16.16 I3C masked interrupt status register (I3C_MISR) 1952
46.16.17 I3C own device characteristics register (I3C_DEVR0) 1953
46.16.18 I3C device x characteristics register (I3C_DEVRx) 1955
46.16.19 I3C maximum read length register (I3C_MAXRLR) 1957
46.16.20 I3C maximum write length register (I3C_MAXWLR) 1958
46.16.21 I3C timing register 0 (I3C_TIMINGR0) 1959
46.16.22 I3C timing register 1 (I3C_TIMINGR1) 1960
46.16.23 I3C timing register 2 (I3C_TIMINGR2) 1962
46.16.24 I3C bus characteristics register (I3C_BCR) 1963
46.16.25 I3C device characteristics register (I3C_DCR) 1964
46.16.26 I3C get capability register (I3C_GETCAPR) 1965
46.16.27 I3C controller-role capability register (I3C_CRCAPR) 1966
46.16.28 I3C get max data speed register (I3C_GETMXDSR) 1967
46.16.29 I3C extended provisioned ID register (I3C_EPIDR) 1969
46.16.30 I3C register map 1970

47 Universal synchronous/asynchronous receiver transmitter (USART/UART) 1973

47.1 Introduction 1973
47.2 USART main features 1973
47.3 USART extended features 1974
47.4 USART implementation 1974
47.5 USART functional description 1976
47.5.1 USART block diagram 1976
47.5.2 USART pins and internal signals 1976
47.5.3 USART clocks 1979
47.5.4 USART character description 1979

47.5.5 USART FIFOs and thresholds . . . 1981
47.5.6 USART transmitter . . . 1981
47.5.7 USART receiver . . . 1984
47.5.8 USART baud rate generation . . . 1991
47.5.9 Tolerance of the USART receiver to clock deviation . . . 1993
47.5.10 USART auto baud rate detection . . . 1994
47.5.11 USART multiprocessor communication . . . 1996
47.5.12 USART Modbus communication . . . 1998
47.5.13 USART parity control . . . 1999
47.5.14 USART LIN (local interconnection network) mode . . . 2000
47.5.15 USART synchronous mode . . . 2002
47.5.16 USART single-wire half-duplex communication . . . 2006
47.5.17 USART receiver timeout . . . 2006
47.5.18 USART smartcard mode . . . 2007
47.5.19 USART IrDA SIR ENDEC block . . . 2011
47.5.20 Continuous communication using USART and DMA . . . 2014
47.5.21 RS232 hardware flow control and RS485 driver enable . . . 2016
47.5.22 USART autonomous mode . . . 2018
47.6 USART in low-power modes . . . 2020
47.7 USART interrupts . . . 2020
47.8 USART registers . . . 2023
47.8.1 USART control register 1 (USART_CR1) . . . 2023
47.8.2 USART control register 1 [alternate] (USART_CR1) . . . 2027
47.8.3 USART control register 2 (USART_CR2) . . . 2030
47.8.4 USART control register 3 (USART_CR3) . . . 2034
47.8.5 USART control register 3 [alternate] (USART_CR3) . . . 2038
47.8.6 USART baud rate register (USART_BRR) . . . 2041
47.8.7 USART guard time and prescaler register (USART_GTPR) . . . 2042
47.8.8 USART receiver timeout register (USART_RTOR) . . . 2043
47.8.9 USART request register (USART_RQR) . . . 2044
47.8.10 USART interrupt and status register (USART_ISR) . . . 2045
47.8.11 USART interrupt and status register [alternate] (USART_ISR) . . . 2051
47.8.12 USART interrupt flag clear register (USART_ICR) . . . 2056
47.8.13 USART receive data register (USART_RDR) . . . 2057
47.8.14 USART transmit data register (USART_TDR) . . . 2058
47.8.15 USART prescaler register (USART_PRESC) . . . 2058
47.8.16 USART autonomous mode control register (USART_AUTOCR) . . . 2059

47.8.17 USART register map 2060

48 Low-power universal asynchronous receiver transmitter (LPUART) 2062

48.1 Introduction 2062
48.2 LPUART main features 2062
48.3 LPUART implementation 2063
48.4 LPUART functional description 2064
48.4.1 LPUART block diagram 2064
48.4.2 LPUART pins and internal signals 2065
48.4.3 LPUART clocks 2067
48.4.4 LPUART character description 2067
48.4.5 LPUART FIFOs and thresholds 2069
48.4.6 LPUART transmitter 2069
48.4.7 LPUART receiver 2073
48.4.8 LPUART baud rate generation 2077
48.4.9 Tolerance of the LPUART receiver to clock deviation 2078
48.4.10 LPUART multiprocessor communication 2079
48.4.11 LPUART parity control 2081
48.4.12 LPUART single-wire half-duplex communication 2082
48.4.13 Continuous communication using DMA and LPUART 2082
48.4.14 RS232 hardware flow control and RS485 driver enable 2085
48.4.15 LPUART autonomous mode 2087
48.5 LPUART in low-power modes 2089
48.6 LPUART interrupts 2090
48.7 LPUART registers 2091
48.7.1 LPUART control register 1 (LPUART_CR1) 2091
48.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 2094
48.7.3 LPUART control register 2 (LPUART_CR2) 2097
48.7.4 LPUART control register 3 (LPUART_CR3) 2099
48.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 2101
48.7.6 LPUART baud rate register (LPUART_BRR) 2103
48.7.7 LPUART request register (LPUART_RQR) 2103
48.7.8 LPUART interrupt and status register (LPUART_ISR) 2104
48.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 2109
48.7.10 LPUART interrupt flag clear register (LPUART_ICR) 2112

48.7.11 LPUART receive data register (LPUART_RDR) 2113
48.7.12 LPUART transmit data register (LPUART_TDR) 2113
48.7.13 LPUART prescaler register (LPUART_PRESC) 2114
48.7.14 LPUART autonomous mode control register (LPUART_AUTOCR) . . 2115
48.7.15 LPUART register map 2115

49 Serial peripheral interface (SPI) 2118

49.1 Introduction 2118
49.2 SPI main features 2118
49.3 SPI implementation 2119
49.4 SPI functional description 2120
49.4.1 SPI block diagram 2120
49.4.2 SPI pins and internal signals 2121
49.4.3 SPI communication general aspects 2123
49.4.4 Communications between one master and one slave 2123
49.4.5 Standard multislave communication 2126
49.4.6 Multimaster communication 2127
49.4.7 Slave select (NSS pin) management 2128
49.4.8 Ready pin (RDY) management 2132
49.4.9 Communication formats 2132
49.4.10 Configuring the SPI 2134
49.4.11 Enabling the SPI 2135
49.4.12 SPI data transmission and reception procedures 2136
49.4.13 Disabling the SPI 2140
49.4.14 Communication using DMA (direct memory addressing) 2141
49.4.15 Autonomous mode 2142
49.5 SPI specific modes and control 2144
49.5.1 TI mode 2144
49.5.2 SPI error flags 2145
49.5.3 CRC computation 2148
49.6 SPI in low-power modes 2149
49.7 SPI interrupts 2149
49.8 SPI registers 2151
49.8.1 SPI control register 1 (SPI_CR1) 2151
49.8.2 SPI control register 2 (SPI_CR2) 2153
49.8.3 SPI configuration register 1 (SPI_CFG1) 2153

49.8.4 SPI configuration register 2 (SPI_CFG2) 2156
49.8.5 SPI interrupt enable register (SPI_IER) 2158
49.8.6 SPI status register (SPI_SR) 2159
49.8.7 SPI interrupt/status flags clear register (SPI_IFCR) 2162
49.8.8 SPI autonomous mode control register (SPI_AUTOCR) 2163
49.8.9 SPI transmit data register (SPI_TXDR) 2163
49.8.10 SPI receive data register (SPI_RXDR) 2164
49.8.11 SPI polynomial register (SPI_CRCPOLY) 2164
49.8.12 SPI transmitter CRC register (SPI_TXCRC) 2165
49.8.13 SPI receiver CRC register (SPI_RXCRC) 2166
49.8.14 SPI underrun data register (SPI_UDRDR) 2166
49.8.15 SPI register map 2167

50 Serial audio interface (SAI) 2168

50.1 Introduction 2168
50.2 SAI main features 2168
50.3 SAI implementation 2169
50.4 SAI functional description 2170
50.4.1 SAI block diagram 2170
50.4.2 SAI pins and internal signals 2171
50.4.3 Main SAI modes 2171
50.4.4 SAI synchronization mode 2172
50.4.5 Audio data size 2173
50.4.6 Frame synchronization 2173
50.4.7 Slot configuration 2176
50.4.8 SAI clock generator 2178
50.4.9 Internal FIFOs 2181
50.4.10 PDM interface 2183
50.4.11 AC’97 link controller 2191
50.4.12 SPDIF output 2192
50.4.13 Specific features 2195
50.4.14 Error flags 2199
50.4.15 Disabling the SAI 2202
50.4.16 SAI DMA interface 2202
50.5 SAI interrupts 2203
50.6 SAI registers 2205

50.6.1 SAI configuration register 1 (SAI_ACR1) 2205
50.6.2 SAI configuration register 2 (SAI_ACR2) 2207
50.6.3 SAI frame configuration register (SAI_AFRCR) 2209
50.6.4 SAI slot register (SAI_ASLOTR) 2210
50.6.5 SAI interrupt mask register (SAI_AIM) 2211
50.6.6 SAI status register (SAI_ASR) 2213
50.6.7 SAI clear flag register (SAI_ACLRFR) 2215
50.6.8 SAI data register (SAI_ADR) 2216
50.6.9 SAI configuration register 1 (SAI_BCR1) 2216
50.6.10 SAI configuration register 2 (SAI_BCR2) 2219
50.6.11 SAI frame configuration register (SAI_BFRCR) 2221
50.6.12 SAI slot register (SAI_BSLOTR) 2222
50.6.13 SAI interrupt mask register (SAI_BIM) 2223
50.6.14 SAI status register (SAI_BSR) 2224
50.6.15 SAI clear flag register (SAI_BCLRFR) 2226
50.6.16 SAI data register (SAI_BDR) 2227
50.6.17 SAI PDM control register (SAI_PDMCR) 2228
50.6.18 SAI PDM delay register (SAI_PDMDLY) 2229
50.6.19 SAI register map 2231

51 FD controller area network (FDCAN) 2233

51.1 Introduction 2233
51.2 FDCAN main features 2235
51.3 FDCAN functional description 2236
51.3.1 FDCAN block diagram 2236
51.3.2 FDCAN pins and internal signals 2237
51.3.3 Bit timing 2238
51.3.4 Operating modes 2239
51.3.5 Error management 2248
51.3.6 Message RAM 2249
51.3.7 FIFO acknowledge handling 2258
51.3.8 FDCAN Rx FIFO element 2258
51.3.9 FDCAN Tx buffer element 2260
51.3.10 FDCAN Tx event FIFO element 2262
51.3.11 FDCAN standard message ID filter element 2263
51.3.12 FDCAN extended message ID filter element 2264
51.4 FDCAN registers 2266

51.4.1 FDCAN core release register (FDCAN_CREL) 2266
51.4.2 FDCAN endian register (FDCAN_ENDN) 2266
51.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 2266
51.4.4 FDCAN test register (FDCAN_TEST) 2267
51.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 2268
51.4.6 FDCAN CC control register (FDCAN_CCCR) 2269
51.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 2270
51.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 2272
51.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2272
51.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 2273
51.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 2274
51.4.12 FDCAN error counter register (FDCAN_ECR) 2274
51.4.13 FDCAN protocol status register (FDCAN_PSR) 2275
51.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 2277
51.4.15 FDCAN interrupt register (FDCAN_IR) 2277
51.4.16 FDCAN interrupt enable register (FDCAN_IE) 2280
51.4.17 FDCAN interrupt line select register (FDCAN_ILS) 2282
51.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 2283
51.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 2283
51.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 2285
51.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 2285
51.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 2286
51.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 2287
51.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 2287
51.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 2288
51.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 2288
51.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 2289
51.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 2289
51.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 2290
51.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 2291
51.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 2291
51.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 2292
51.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 2292
51.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 2293
51.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 2293
51.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 2294

51.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 2294
51.4.38 FDCAN register map 2295

52 Universal serial bus full-speed host/device interface (USB) 2299

52.1 Introduction 2299
52.2 USB main features 2299
52.3 USB implementation 2299
52.4 USB functional description 2300
52.4.1 USB block diagram 2300
52.4.2 USB pins and internal signals 2300
52.4.3 USB reset and clocks 2301
52.4.4 General description and Device mode functionality 2301
52.4.5 Description of USB blocks used in both Device and Host modes . . . 2302
52.4.6 Description of host frame scheduler (HFS) specific to Host mode . . 2303
52.5 Programming considerations for Device and Host modes 2304
52.5.1 Generic USB Device programming 2304
52.5.2 System and power-on reset 2305
52.5.3 Double-buffered endpoints and usage in Device mode 2312
52.5.4 Double buffered channels: usage in Host mode 2314
52.5.5 Isochronous transfers in Device mode 2315
52.5.6 Isochronous transfers in Host mode 2316
52.5.7 Suspend/resume events 2317
52.6 USB registers 2321
52.6.1 USB control register (USB_CNTR) 2321
52.6.2 USB interrupt status register (USB_ISTR) 2324
52.6.3 USB frame number register (USB_FNR) 2328
52.6.4 USB Device address (USB_DADDR) 2328
52.6.5 USB LPM control and status register (USB_LPMCSR) 2329
52.6.6 USB battery charging detector (USB_BCDR) 2330
52.6.7 USB endpoint/channel n register (USB_CHEPnR) 2331
52.6.8 USB register map 2340
52.7 USBSRAM registers 2341
52.7.1 Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n) 2342
52.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n) 2342

52.7.3 Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n) . . . 2344
52.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n) . . . 2345
52.7.5 USBSRAM register map . . . 2346

53 Debug support (DBG) . . . 2347
53.1 Introduction . . . 2347
53.2 DBG functional description . . . 2348
53.2.1 DBG block diagram . . . 2348
53.2.2 DBG pins and internal signals . . . 2348
53.2.3 DBG reset and clocks . . . 2349
53.2.4 DBG power domains . . . 2349
53.2.5 Debug and low-power modes . . . 2349
53.2.6 Security . . . 2350
53.3 Serial-wire and JTAG debug port (SWJ-DP) . . . 2351
53.3.1 JTAG debug port . . . 2352
53.3.2 Serial-wire debug port . . . 2354
53.3.3 Debug port registers . . . 2355
53.3.4 Debug port register map . . . 2361
53.4 Access ports . . . 2362
53.4.1 Access port registers . . . 2362
53.4.2 Access port register map . . . 2367
53.5 ROM tables . . . 2368
53.5.1 MCU ROM table registers . . . 2368
53.5.2 MCU ROM table register map . . . 2372
53.5.3 Processor ROM table registers . . . 2373
53.5.4 Processor ROM table register map . . . 2376
53.6 Data watchpoint and trace unit (DWT) . . . 2378
53.6.1 DWT registers . . . 2378
53.6.2 DWT register map . . . 2390
53.7 Instrumentation trace macrocell (ITM) . . . 2393
53.7.1 ITM registers . . . 2393
53.7.2 ITM register map . . . 2400
53.8 Breakpoint unit (BPU) . . . 2401
53.8.1 BPU registers . . . 2401
53.8.2 BPU register map . . . 2406

53.9 Embedded Trace Macrocell (ETM) 2408
53.9.1 ETM registers 2408
53.9.2 ETM register map 2430
53.10 Trace port interface unit (TPIU) 2434
53.10.1 TPIU registers 2434
53.10.2 TPIU register map 2443
53.11 Cross-trigger interface (CTI) 2445
53.11.1 CTI registers 2445
53.11.2 CTI register map 2455
53.12 Microcontroller debug unit (DBGMCU) 2457
53.12.1 DBGMCU registers 2457
53.12.2 DBGMCU register map 2466
53.13 References 2468

54 Device electronic signature 2469
54.1 Unique device ID register (96 bits) 2469
54.2 Flash memory size data register 2470
54.3 Package data register 2471

55 Important security notice 2472

56 Revision history 2473

List of tables

Table 1. Example of memory map security attribution versus SAU configuration regions 92
Table 2. Securable peripherals by TZSC 93
Table 3. TrustZone aware peripherals 95
Table 4. Memory map and peripheral register boundary addresses 99
Table 5. Configuring security attributes with IDAU and SAU 110
Table 6. MPCBBx resources 112
Table 7. DMA channel use (security). 115
Table 8. Secure alternate function between peripherals and allocated I/Os 117
Table 9. Nonsecure peripherals not connectable to secure I/Os. 118
Table 10. Nonsecure peripherals connectable to secure I/Os 118
Table 11. TrustZone-aware DBGMCU access management 120
Table 12. DMA channel use (privilege). 124
Table 13. Accelerated-cryptographic operations 130
Table 14. Main product life-cycle transitions. 131
Table 15. Typical product life-cycle phases 132
Table 16. OEM1/2 RDP unlocking methods 134
Table 17. Debug protection with RDP 135
Table 18. Software intellectual property protection with RDP. 136
Table 19. Boot modes when TrustZone is disabled (TZEN = 0). 138
Table 20. Boot modes when TrustZone is enabled (TZEN = 1) 139
Table 21. Boot space versus RDP protection. 139
Table 22. GTZC features 143
Table 23. GTZC1 subblocks address offset 144
Table 24. MPCBB resource assignment 144
Table 25. GTZC interrupt request 147
Table 26. GTZC1 TZSC register map and reset values 159
Table 27. GTZC1 TZIC register map and reset values. 179
Table 28. GTZC1 MPCBB1 register map and reset values 183
Table 29. GTZC1 MPCBB2 register map and reset values 186
Table 30. Internal SRAM features 188
Table 31. Effect of low-power modes on RAMCFG 189
Table 32. RAMCFG interrupt requests 189
Table 33. RAMCFG register map and reset values 195
Table 34. Flash module 1-Mbyte dual bank-organization. 198
Table 35. Flash module 512-Kbyte dual bank-organization for N/A 199
Table 36. Flash module 1-Mbyte dual bank-organization for STM32U375/385 199
Table 37. Number of wait states according to CPU clock (HCLK) frequency (LPM = 0) 200
Table 38. Number of wait states according to CPU clock (HCLK) frequency (LPM = 1) 200
Table 39. Flash operation interrupted by a system reset 209
Table 40. User option-byte organization mapping 210
Table 41. Default secure option bytes after TZEN activation 213
Table 42. Secure watermark-based area 214
Table 43. Secure hide protection 216
Table 44. Secure and HDP protections 216
Table 45. Write access register to HDPx_EXT and HDPxEXT_ACCDIS. 217
Table 46. HDP extension protections 218
Table 47. Flash security state 219
Table 48. WRP protection 223

Table 49. Flash memory readout protection status (TZEN = 0) . . . 224
Table 50. Access status versus protection level and execution mode when TZEN = 0 . . . 225
Table 51. Flash memory readout protection status (TZEN = 1) . . . 225
Table 52. Access status versus protection level and execution modes when TZEN = 1 . . . 227
Table 53. Macros for RSS services . . . 232
Table 54. RSS lib interface functions . . . 233
Table 55. Flash memory access versus RDP level when TrustZone is active (TZEN = 1) . . . 234
Table 56. Flash memory access versus HDPx and HDPx extended (x = 1 or 2) . . . 234
Table 57. Flash memory access versus RDP level when TrustZone is disabled (TZEN = 0) . . . 235
Table 58. Flash memory mass erase versus RDP level when TrustZone is active (TZEN = 1) . . . 235
Table 59. Flash system memory, OTP and RSS accesses (valid for all RDP levels) . . . 235
Table 60. FLASH registers access . . . 236
Table 61. FLASH page access versus privilege mode . . . 236
Table 62. FLASH mass erase versus privilege mode . . . 236
Table 63. FLASH_SECBByRx registers access when TrustZone is active (TZEN = 1) . . . 237
Table 64. FLASH_PRIVBByRx registers access when TrustZone is active (TZEN = 1) . . . 237
Table 65. FLASH_PRIVBByRx registers access when TrustZone is disabled (TZEN = 0) . . . 237
Table 66. Flash interrupt requests . . . 237
Table 67. FLASH register map and reset values . . . 273
Table 68. ICACHE features . . . 278
Table 69. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 280
Table 70. TAG memory dimensioning parameters for direct-mapped cache mode . . . 281
Table 71. ICACHE cacheability for AHB transaction . . . 283
Table 72. Memory configurations . . . 283
Table 73. ICACHE remap region size, base address, and remap address . . . 284
Table 74. ICACHE interrupts . . . 288
Table 75. ICACHE register map and reset values . . . 293
Table 76. PWR input/output pins . . . 294
Table 77. PWR internal input/output signals . . . 295
Table 78. PWR wake-up source selection . . . 295
Table 79. PVM features . . . 304
Table 80. Low-power mode summary . . . 308
Table 81. Functionalities depending on the working mode . . . 309
Table 82. Sleep mode . . . 313
Table 83. Stop 0 mode . . . 315
Table 84. Stop 1 mode . . . 317
Table 85. Stop 2 mode . . . 319
Table 86. Stop 3 mode . . . 322
Table 87. Standby mode . . . 325
Table 88. Shutdown mode . . . 327
Table 89. Power modes output states versus MCU power modes . . . 328
Table 90. PWR Security configuration summary . . . 329
Table 91. PWR interrupt requests . . . 330
Table 92. PWR register map and reset values . . . 354
Table 93. RCC input/output signals connected to package pins or balls . . . 357
Table 94. HSE/LSE clock sources . . . 362
Table 95. MSIS and MSIK ranges per internal MSIRCs (PLL mode disabled) . . . 364
Table 96. MSIS and MSIK frequencies of MSIRC1 in PLL mode . . . 366
Table 97. Bus maximum frequency . . . 369
Table 98. Clock source maximum frequency . . . 369

Table 99. RCC security configuration summary . . . 375
Table 100. Interrupt sources and control . . . 378
Table 101. RCC register map and reset values . . . 437
Table 102. CRS features . . . 443
Table 103. CRS internal input/output signals . . . 444
Table 104. CRS interconnection . . . 445
Table 105. Effect of low-power modes on CRS . . . 448
Table 106. Interrupt control bits . . . 448
Table 107. CRS register map and reset values . . . 453
Table 108. Port bit configuration . . . 455
Table 109. GPIO secured bits . . . 463
Table 110. Effect of low-power modes on the GPIO . . . 464
Table 111. GPIO register map and reset values . . . 474
Table 112. TrustZone security and privilege register accesses . . . 478
Table 113. BOOSTEN and ANASWVDD set/reset . . . 479
Table 114. SYSCFG register map and reset values . . . 488
Table 115. Peripherals interconnect matrix . . . 490
Table 116. GPDMA1 channel implementation . . . 506
Table 117. GPDMA1 wake-up in low-power modes . . . 506
Table 118. Programmed GPDMA1 request . . . 506
Table 119. Programmed GPDMA1 request as a block request . . . 509
Table 120. GPDMA1 channel with peripheral early termination . . . 510
Table 121. Programmed GPDMA1 request with peripheral early termination . . . 510
Table 122. Programmed GPDMA1 trigger . . . 510
Table 123. Programmed GPDMA source/destination burst . . . 531
Table 124. Programmed data handling . . . 536
Table 125. Effect of low-power modes on GPDMA . . . 549
Table 126. GPDMA interrupt requests . . . 550
Table 127. GPDMA register map and reset values . . . 578
Table 128. STM32U3 series vector table . . . 581
Table 129. EXTI signals . . . 588
Table 130. EVG signals . . . 588
Table 131. EXTI line connections . . . 589
Table 132. Masking functionality . . . 591
Table 133. Register protection overview . . . 592
Table 134. EXTI register map and reset values . . . 601
Table 135. CRC internal input/output signals . . . 604
Table 136. CRC register map and reset values . . . 609
Table 137. OCTOSPI implementation . . . 611
Table 138. OCTOSPI input/output pins . . . 613
Table 139. OCTOSPI internal signals . . . 613
Table 140. Command/address phase description . . . 622
Table 141. OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM) . . . 632
Table 142. Address alignment cases . . . 639
Table 143. OCTOSPI interrupt requests . . . 640
Table 144. OCTOSPI register map and reset values . . . 663
Table 145. SDMMC features . . . 667
Table 146. SDMMC operation modes SD and SDIO . . . 670
Table 147. SDMMC operation modes e•MMC . . . 670
Table 148. SDMMC internal input/output signals . . . 671
Table 149. SDMMC pins . . . 672

Table 150. SDMMC Command and data phase selection . . . 673
Table 151. Command token format . . . 679
Table 152. Short response with CRC token format . . . 680
Table 153. Short response without CRC token format . . . 680
Table 154. Long response with CRC token format . . . 680
Table 155. Specific Commands overview . . . 681
Table 156. Command path status flags . . . 682
Table 157. Command path error handling . . . 682
Table 158. Data token format . . . 690
Table 159. Data path status flags and clear bits . . . 690
Table 160. Data path error handling . . . 692
Table 161. Data FIFO access . . . 693
Table 162. Transmit FIFO status flags . . . 694
Table 163. Receive FIFO status flags . . . 695
Table 164. AHB and SDMMC_CK clock frequency relation . . . 700
Table 165. SDIO special operation control . . . 700
Table 166. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 704
Table 167. CMD12 use cases . . . 709
Table 168. SDMMC interrupts . . . 723
Table 169. Response type and SDMMC_RESPxR registers . . . 730
Table 170. SDMMC register map . . . 746
Table 171. STM32U3 series features . . . 749
Table 172. DLYB internal input/output signals . . . 750
Table 173. Delay block control . . . 751
Table 174. DLYB register map and reset values . . . 754
Table 175. ADC features . . . 757
Table 176. ADC input/output pins . . . 759
Table 177. ADC internal input/output signals . . . 759
Table 178. ADC1/2 interconnection . . . 759
Table 179. Configuring the trigger polarity for regular external triggers . . . 777
Table 180. Configuring the trigger polarity for injected external triggers . . . 777
Table 181. TSAR timings depending on resolution . . . 790
Table 182. Offset computation versus data resolution . . . 794
Table 183. 12-bit data formats . . . 796
Table 184. Numerical examples for 32-bit or 16-bit format . . . 797
Table 185. Analog watchdog channel selection . . . 807
Table 186. Analog watchdog 1, 2, 3 comparison . . . 808
Table 187. Maximum output results versus N and M . . . 812
Table 188. Oversampler operating mode summary . . . 816
Table 189. DELAY bits versus ADC resolution . . . 822
Table 190. Effect of low-power modes on the ADC . . . 838
Table 191. ADC interrupts . . . 839
Table 192. ADC register map and reset values for each ADC . . . 874
Table 193. ADC register map and reset values (master and slave ADC common registers) . . . 876
Table 194. DAC features . . . 879
Table 195. DAC input/output pins . . . 881
Table 196. DAC internal input/output signals . . . 881
Table 197. DAC interconnection . . . 882
Table 198. Data format (case of 12-bit data) . . . 884
Table 199. HFSEL description . . . 885
Table 200. Sample and refresh timings . . . 890

Table 201. Channel output modes summary . . . 892
Table 202. Effect of low-power modes on DAC . . . 899
Table 203. DAC interrupts . . . 900
Table 204. DAC register map and reset values . . . 916
Table 205. VREFBUF typical values . . . 919
Table 206. VREF buffer modes . . . 920
Table 207. VREFBUF register map and reset values . . . 922
Table 208. COMP1 noninverting input assignment . . . 924
Table 209. COMP1 inverting input assignment . . . 925
Table 210. COMP2 noninverting input assignment . . . 925
Table 211. COMP2 inverting input assignment . . . 925
Table 212. COMP1 output-blanking PWM assignment . . . 925
Table 213. COMP2 output-blanking PWM assignment . . . 926
Table 214. Comparator behavior in the low-power modes . . . 929
Table 215. Interrupt control bits . . . 930
Table 216. COMP register map and reset values . . . 933
Table 217. Operational amplifier possible connections . . . 935
Table 218. Operating modes and calibration . . . 940
Table 219. Effect of low-power modes on the OPAMP . . . 941
Table 220. OPAMP register map and reset values . . . 946
Table 221. ADF features . . . 948
Table 222. ADF external pins . . . 949
Table 223. ADF internal signals . . . 949
Table 224. ADF trigger connections . . . 950
Table 225. MDF ADC data connections . . . 950
Table 226. Control of the common clock generation . . . 957
Table 227. Clock constraints with respect to the incoming stream . . . 958
Table 228. Data size according to CIC order and CIC decimation values . . . 963
Table 229. Possible gain values . . . 964
Table 230. Recommended maximum gain values versus CIC decimation ratios . . . 966
Table 231. Most common microphone settings . . . 966
Table 232. HPF 3 dB cut-off frequency examples . . . 968
Table 233. ANSLP values versus FRSIZE and sampling rates . . . 981
Table 234. Threshold values according SNTHR . . . 982
Table 235. Register protection summary . . . 988
Table 236. Effect of low-power modes on ADF . . . 989
Table 237. ADF interrupt requests . . . 990
Table 238. Examples of ADF settings for microphone capture . . . 991
Table 239. Programming sequence (CIC4) . . . 992
Table 240. Programming sequence (CIC5) . . . 993
Table 241. Output signal levels . . . 998
Table 242. ADF register map and reset values . . . 1020
Table 243. Acquisition sequence summary . . . 1026
Table 244. Spread spectrum deviation versus AHB clock frequency . . . 1028
Table 245. I/O state depending on its mode and IODEF bit value . . . 1029
Table 246. Effect of low-power modes on TSC . . . 1031
Table 247. Interrupt control bits . . . 1031
Table 248. TSC register map and reset values . . . 1039
Table 249. Peripherals connected by CCB interconnect . . . 1042
Table 250. CCB internal input/output signals . . . 1043
Table 251. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0) . . . 1049

Table 252. Blob use steps for ECC signature (CCOP = 0b11000011) . . . 1051
Table 253. Blob use steps for signing public key computation (CCOP = 0b10000001) . . . 1053
Table 254. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0) . . . 1055
Table 255. Blob use steps for scalar multiplication (CCOP = 0b10000001) . . . 1056
Table 256. Blob creation steps for modular exponentiation (CCOP = 0b01000100) . . . 1057
Table 257. Blob use steps for modular exponentiation (CCOP = 0b01000101) . . . 1058
Table 258. CCB register map and reset values . . . 1064
Table 259. RNG internal input/output signals . . . 1066
Table 260. RNG interrupt requests . . . 1074
Table 261. RNG configurations . . . 1075
Table 262. Additional health test configurations . . . 1075
Table 263. Configuration selection . . . 1076
Table 264. RNG register map and reset map . . . 1081
Table 265. AES versus SAES features . . . 1083
Table 266. AES internal input/output signals . . . 1084
Table 267. AES approved symmetric key functions . . . 1084
Table 268. Counter mode initialization vector definition . . . 1094
Table 269. Initialization of IV registers in GCM mode . . . 1096
Table 270. GCM last block definition . . . 1096
Table 271. Initialization of IV registers in CCM mode . . . 1103
Table 272. AES data swapping example . . . 1106
Table 273. Key endianness in AES_KEYRx registers (128/256-bit keys) . . . 1108
Table 274. IVI bitfield spread over AES_IVRx registers . . . 1109
Table 275. AES interrupt requests . . . 1110
Table 276. Processing latency for ECB, CBC and CTR . . . 1111
Table 277. Processing latency for GCM and CCM (in clock cycles) . . . 1112
Table 278. AES register map and reset values . . . 1124
Table 279. AES versus SAES features . . . 1128
Table 280. SAES internal input/output signals . . . 1129
Table 281. SAES approved symmetric key functions . . . 1130
Table 282. Counter mode initialization vector definition . . . 1140
Table 283. Initialization of IV registers in GCM mode . . . 1143
Table 284. GCM last block definition . . . 1143
Table 285. Initialization of IV registers in CCM mode . . . 1149
Table 286. AES data swapping example . . . 1158
Table 287. Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . 1160
Table 288. IVI bitfield spread over SAES_IVRx registers . . . 1162
Table 289. SAES interrupt requests . . . 1164
Table 290. Processing latency for ECB, CBC and CTR . . . 1165
Table 291. Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . 1165
Table 292. SAES register map and reset values . . . 1180
Table 293. HASH internal input/output signals . . . 1183
Table 294. Information on supported hash algorithms . . . 1184
Table 295. Hash processor outputs . . . 1187
Table 296. Processing time (in clock cycle) . . . 1193
Table 297. HASH interrupt requests . . . 1194
Table 298. HASH1 register map and reset values . . . 1202
Table 299. Internal input/output signals . . . 1205
Table 300. PKA integer arithmetic functions list . . . 1206
Table 301. PKA prime field (Fp) elliptic curve functions list . . . 1207
Table 302. Chaining mode to/from PKA . . . 1212
Table 303. Integrity protected parameters list . . . 1214

Table 304. Confidentiality protected parameter list . . . 1214
Table 305. Montgomery parameter computation . . . 1217
Table 306. Modular addition . . . 1217
Table 307. Modular subtraction . . . 1218
Table 308. Montgomery multiplication . . . 1219
Table 309. Modular exponentiation (normal mode) . . . 1219
Table 310. Modular exponentiation (fast mode) . . . 1220
Table 311. Modular exponentiation (protected mode) . . . 1220
Table 312. Modular inversion . . . 1221
Table 313. Modular reduction . . . 1221
Table 314. Arithmetic addition . . . 1221
Table 315. Arithmetic subtraction . . . 1222
Table 316. Arithmetic multiplication . . . 1222
Table 317. Arithmetic comparison . . . 1222
Table 318. CRT exponentiation . . . 1223
Table 319. Point on elliptic curve Fp check . . . 1224
Table 320. ECC Fp scalar multiplication . . . 1224
Table 321. ECDSA sign - Inputs . . . 1226
Table 322. ECDSA sign - Outputs . . . 1226
Table 323. Extended ECDSA sign - Extra outputs . . . 1227
Table 324. ECDSA verification - Inputs . . . 1227
Table 325. ECDSA verification - Outputs . . . 1228
Table 326. ECC complete addition . . . 1228
Table 327. ECC double base ladder . . . 1229
Table 328. ECC projective to affine . . . 1230
Table 329. Family of supported curves for ECC operations . . . 1231
Table 330. Modular exponentiation . . . 1232
Table 331. ECC scalar multiplication . . . 1232
Table 332. ECDSA signature average computation time . . . 1233
Table 333. ECDSA verification average computation times . . . 1233
Table 334. ECC double base ladder average computation times . . . 1233
Table 335. ECC projective to affine average computation times . . . 1233
Table 336. ECC complete addition average computation times . . . 1233
Table 337. Point on elliptic curve Fp check average computation times . . . 1233
Table 338. Montgomery parameters average computation times . . . 1234
Table 339. PKA interrupt requests . . . 1234
Table 340. PKA register map and reset values . . . 1241
Table 341. TIM input/output pins . . . 1245
Table 342. TIM internal input/output signals . . . 1245
Table 343. Interconnect to the tim_ti1 input multiplexer . . . 1246
Table 344. Interconnect to the tim_ti2 input multiplexer . . . 1247
Table 345. Interconnect to the tim_ti3 input multiplexer . . . 1247
Table 346. Interconnect to the tim_ti4 input multiplexer . . . 1247
Table 347. Internal trigger connection . . . 1247
Table 348. Interconnect to the tim_etr input multiplexer . . . 1248
Table 349. Timer break interconnect . . . 1248
Table 350. Timer break2 interconnect . . . 1248
Table 351. System break interconnect . . . 1249
Table 352. Interconnect to the ocref_clr input multiplexer . . . 1249
Table 353. CCR and ARR register change dithering pattern . . . 1282
Table 354. CCR register change dithering pattern in center-aligned PWM mode . . . 1283
Table 355. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 1295

Table 356. Break protection disarming conditions . . . 1297
Table 357. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1306
Table 358. Counting direction versus encoder signals and polarity settings . . . 1310
Table 359. DMA request . . . 1332
Table 360. Effect of low-power modes on TIM1 . . . 1333
Table 361. Interrupt requests . . . 1333
Table 362. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 1360
Table 363. TIM1 register map and reset values . . . 1383
Table 364. STM32U3 series general purpose timers . . . 1387
Table 365. TIM input/output pins . . . 1389
Table 366. TIM internal input/output signals . . . 1389
Table 367. Interconnect to the tim_ti1 input multiplexer . . . 1390
Table 368. Interconnect to the tim_ti2 input multiplexer . . . 1390
Table 369. Interconnect to the tim_ti3 input multiplexer . . . 1390
Table 370. Interconnect to the tim_ti4 input multiplexer . . . 1391
Table 371. TIMx internal trigger connection . . . 1391
Table 372. Interconnect to the tim_etr input multiplexer . . . 1391
Table 373. Interconnect to the tim_ocref_clr input multiplexer . . . 1392
Table 374. CCR and ARR register change dithering pattern . . . 1423
Table 375. CCR register change dithering pattern in center-aligned PWM mode . . . 1424
Table 376. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 1433
Table 377. Counting direction versus encoder signals and polarity settings . . . 1438
Table 378. DMA request . . . 1464
Table 379. Effect of low-power modes on TIM2/TIM3/TIM4 . . . 1464
Table 380. Interrupt requests . . . 1465
Table 381. Output control bit for standard tim_ocx channels . . . 1486
Table 382. TIM2/TIM3/TIM4 register map and reset values . . . 1498
Table 383. TIM internal input/output signals . . . 1502
Table 384. TIMx_ARR register change dithering pattern . . . 1512
Table 385. DMA request . . . 1514
Table 386. Effect of low-power modes on TIM6/TIM7 . . . 1514
Table 387. Interrupt request . . . 1514
Table 388. TIMx register map and reset values . . . 1521
Table 389. TIM input/output pins . . . 1525
Table 390. TIM internal input/output signals . . . 1526
Table 391. Interconnect to the tim_ti1 input multiplexer . . . 1526
Table 392. Interconnect to the tim_ti2 input multiplexer . . . 1527
Table 393. TIMx internal trigger connection . . . 1527
Table 394. Timer break interconnect . . . 1528
Table 395. System break interconnect . . . 1528
Table 396. Interconnect to the ocref_clr input multiplexer . . . 1528
Table 397. CCR and ARR register change dithering pattern . . . 1548
Table 398. Break protection disarming conditions . . . 1557
Table 399. DMA request . . . 1570
Table 400. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 1570
Table 401. Interrupt requests . . . 1571
Table 402. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 1588
Table 403. TIM15 register map and reset values . . . 1601
Table 404. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 1614

Table 405. TIM16/TIM17 register map and reset values . . . 1628
Table 406. STM32U3 series LPTIM features . . . 1631
Table 407. LPTIM1/2/3 input/output pins . . . 1633
Table 408. LPTIM4 input/output pins . . . 1633
Table 409. LPTIM1/2/3 internal signals . . . 1634
Table 410. LPTIM4 internal signals . . . 1634
Table 411. LPTIM1/2/3/4 external trigger connections . . . 1635
Table 412. LPTIM1/2/3/4 input 1 connections . . . 1635
Table 413. LPTIM1/2 input 2 connections . . . 1635
Table 414. LPTIM1 input capture 1 connections . . . 1635
Table 415. LPTIM2/3 input capture 1 connections . . . 1636
Table 416. LPTIM1 input capture 2 connections . . . 1636
Table 417. LPTIM2 input capture 2 connections . . . 1636
Table 418. LPTIM3 input capture 2 connections . . . 1636
Table 419. Prescaler division ratios . . . 1638
Table 420. Encoder counting scenarios . . . 1645
Table 421. Input capture Glitch filter latency (in counter step unit) . . . 1649
Table 422. Effect of low-power modes on the LPTIM . . . 1654
Table 423. Interrupt events . . . 1655
Table 424. LPTIM register map and reset values . . . 1679
Table 425. STM32U3 series IWDG features . . . 1684
Table 426. IWDG delays versus actions . . . 1685
Table 427. IWDG internal input/output signals . . . 1685
Table 428. Effect of low power modes on IWDG . . . 1690
Table 429. IWDG interrupt request . . . 1692
Table 430. IWDG register map and reset values . . . 1697
Table 431. WWDG features . . . 1698
Table 432. WWDG internal input/output signals . . . 1699
Table 433. WWDG interrupt requests . . . 1702
Table 434. WWDG register map and reset values . . . 1704
Table 435. RTC input/output pins . . . 1708
Table 436. RTC internal input/output signals . . . 1708
Table 437. RTC interconnection . . . 1709
Table 438. RTC_OUT mapping . . . 1709
Table 439. Effect of low-power modes on RTC . . . 1725
Table 440. RTC pins functionality over modes . . . 1725
Table 441. Nonsecure interrupt requests . . . 1726
Table 442. Secure interrupt requests . . . 1726
Table 443. RTC register map and reset values . . . 1759
Table 444. TAMP input/output pins . . . 1764
Table 445. TAMP internal input/output signals . . . 1764
Table 446. TAMP interconnection . . . 1765
Table 447. PC13 pin configuration . . . 1767
Table 448. Backup registers and BHK tamper protection . . . 1771
Table 449. Other device resource x tamper protection . . . 1771
Table 450. Effect of low-power modes on TAMP . . . 1775
Table 451. TAMP pins functionality over modes . . . 1775
Table 452. Interrupt requests . . . 1775
Table 453. TAMP register map and reset values . . . 1789
Table 454. I2C implementation . . . 1793
Table 455. I2C input/output pins . . . 1794
Table 456. I2C internal input/output signals . . . 1795

Table 457. Trigger interconnections . . . 1795
Table 458. Comparison of analog and digital filters . . . 1797
Table 459. I²C-bus and SMBus specification data setup and hold times . . . 1799
Table 460. I2C configuration . . . 1803
Table 461. I²C-bus and SMBus specification clock timings . . . 1814
Table 462. Timing settings for f_{I2CCLK} of 8 MHz . . . 1824
Table 463. Timing settings for f_{I2CCLK} of 16 MHz . . . 1824
Table 464. Timing settings for f_{I2CCLK} of 48 MHz . . . 1825
Table 465. SMBus timeout specifications . . . 1827
Table 466. SMBus with PEC configuration . . . 1829
Table 467. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 1830
Table 468. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 1830
Table 469. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 1830
Table 470. Effect of low-power modes to I2C . . . 1840
Table 471. I2C interrupt requests . . . 1841
Table 472. I2C register map and reset values . . . 1858
Table 473. I3C instantiation . . . 1862
Table 474. I3C wake-up . . . 1862
Table 475. I3C autonomous mode with DMA in Stop mode(s) . . . 1862
Table 476. I3C FIFOs implementation . . . 1863
Table 477. I3C triggers, when I3C as controller . . . 1863
Table 478. I3C interrupt(s) . . . 1864
Table 479. I3C peripheral controller/target features versus MIPI v1.1 . . . 1864
Table 480. I3C input/output pins . . . 1865
Table 481. I3C internal input/output signals . . . 1865
Table 482. I3C register usage . . . 1877
Table 483. I3C registers/fields usage versus controller state . . . 1878
Table 484. I3C registers/fields usage versus target state . . . 1881
Table 485. List of supported I3C CCCs, as controller/target . . . 1884
Table 486. I3C controller error management . . . 1916
Table 487. I3C target error management . . . 1918
Table 488. Effect of low-power modes . . . 1922
Table 489. I3C interrupt requests . . . 1923
Table 490. I3C register map and reset values . . . 1970
Table 491. Instance implementation on STM32U3 series . . . 1974
Table 492. USART/LPUART features . . . 1974
Table 493. USART/UART input/output pins . . . 1977
Table 494. USART internal input/output signals . . . 1978
Table 495. USART interconnection (USART1/3 and UART4/5) . . . 1978
Table 496. Noise detection from sampled data . . . 1990
Table 497. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 1994
Table 498. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 1994
Table 499. USART frame formats . . . 1999
Table 500. Effect of low-power modes on the USART . . . 2020
Table 501. USART interrupt requests . . . 2021
Table 502. USART register map and reset values . . . 2060
Table 503. Instance implementation on STM32U3 series . . . 2063
Table 504. USART/LPUART features . . . 2063
Table 505. LPUART input/output pins . . . 2065
Table 506. LPUART internal input/output signals . . . 2065
Table 507. LPUART interconnections (LPUART1) . . . 2066
Table 508. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 2077

Table 509. Tolerance of the LPUART receiver . . . 2078
Table 511. Effect of low-power modes on the LPUART . . . 2089
Table 512. LPUART interrupt requests . . . 2090
Table 513. LPUART register map and reset values . . . 2115
Table 514. SPI features . . . 2119
Table 515. SPI input/output pins . . . 2121
Table 516. SPI internal input/output signals . . . 2122
Table 517. SPI interconnection (SPI1 and SPI2) . . . 2122
Table 518. SPI interconnection (SPI3) . . . 2123
Table 519. Effect of low-power modes on the SPI . . . 2149
Table 520. SPI wake-up and interrupt requests . . . 2150
Table 521. SPI register map and reset values . . . 2167
Table 522. SAI features . . . 2169
Table 523. SAI internal input/output signals . . . 2171
Table 524. SAI input/output pins . . . 2171
Table 525. MCLK_x activation conditions . . . 2178
Table 526. Clock generator programming examples . . . 2181
Table 527. SAI_A configuration for TDM mode . . . 2188
Table 528. TDM frame configuration examples . . . 2190
Table 529. SOPD pattern . . . 2193
Table 530. Parity bit calculation . . . 2193
Table 531. Audio sampling frequency versus symbol rates . . . 2194
Table 532. SAI interrupt sources . . . 2203
Table 533. SAI register map and reset values . . . 2231
Table 534. CAN subsystem I/O signals . . . 2237
Table 535. CAN subsystem I/O pins . . . 2237
Table 536. DLC coding in FDCAN . . . 2241
Table 537. Possible configurations for frame transmission . . . 2255
Table 538. Rx FIFO element . . . 2258
Table 539. Rx FIFO element description . . . 2258
Table 540. Tx buffer and FIFO element . . . 2260
Table 541. Tx buffer element description . . . 2260
Table 542. Tx event FIFO element . . . 2262
Table 543. Tx event FIFO element description . . . 2262
Table 544. Standard message ID filter element . . . 2263
Table 545. Standard message ID filter element field description . . . 2264
Table 546. Extended message ID filter element . . . 2264
Table 547. Extended message ID filter element field description . . . 2265
Table 548. FDCAN register map and reset values . . . 2295
Table 549. STM32U3 USB implementation . . . 2299
Table 550. USB input/output pins . . . 2300
Table 551. Double-buffering buffer flag definition . . . 2313
Table 552. Bulk double-buffering memory buffers usage (Device mode) . . . 2313
Table 553. Bulk double-buffering memory buffers usage (Host mode) . . . 2315
Table 554. Isochronous memory buffers usage . . . 2316
Table 555. Isochronous memory buffers usage . . . 2317
Table 556. Resume event detection . . . 2319
Table 557. Resume event detection for host . . . 2320
Table 558. Reception status encoding . . . 2338
Table 559. Endpoint/channel type encoding . . . 2338
Table 560. Endpoint/channel kind meaning . . . 2338
Table 561. Transmission status encoding . . . 2338

Table 562. USB register map and reset values . . . 2340
Table 563. Definition of allocated buffer memory . . . 2343
Table 564. USBSRAM register map and reset values . . . 2346
Table 565. JTAG/Serial-wire debug port pins . . . 2348
Table 566. Trace port pins . . . 2348
Table 567. Single-wire trace port pins . . . 2349
Table 568. Authentication signal states . . . 2350
Table 569. JTAG-DP data registers . . . 2353
Table 570. Packet request . . . 2354
Table 571. ACK response . . . 2355
Table 572. Data transfer . . . 2355
Table 573. Debug port register map and reset values . . . 2361
Table 574. Access port register map and reset values . . . 2367
Table 575. MCU ROM table register map and reset values . . . 2372
Table 576. CPU ROM table register map and reset values . . . 2376
Table 577. DWT register map and reset values . . . 2390
Table 578. ITM register map and reset values . . . 2400
Table 579. BPU register map and reset values . . . 2406
Table 580. ETM register map and reset values . . . 2430
Table 581. TPIU register map and reset values . . . 2443
Table 582. CTI register map and reset values . . . 2455
Table 583. DBGMCU register map and reset values . . . 2466
Table 584. Document revision history . . . 2473

List of figures

Figure 1. System architecture . . . 90
Figure 2. Memory map based on IDAU mapping . . . 98
Figure 3. Secure/nonsecure partitioning using TrustZone technology . . . 107
Figure 4. Sharing memory map between CPU in secure and nonsecure state . . . 109
Figure 5. Secure world transition and memory partitioning . . . 109
Figure 6. Global TrustZone framework and TrustZone awareness . . . 111
Figure 7. Flash memory TrustZone protections . . . 114
Figure 8. Flash memory secure HDP area . . . 121
Figure 9. Key management principle . . . 128
Figure 10. Device life-cycle security . . . 131
Figure 11. RDP level transition scheme . . . 133
Figure 12. Collaborative development principle . . . 136
Figure 13. GTZC in Armv8-M subsystem block diagram . . . 143
Figure 14. GTZC block diagram . . . 145
Figure 15. MPCBB block diagram . . . 146
Figure 16. Secure, HDP, and HDP extension areas . . . 218
Figure 17. Flash memory security attributes and protections in case of no bank swap (SWAP_BANK = 0) . . . 221
Figure 18. Flash memory security attributes and protections in case of bank swap (SWAP_BANK = 1) . . . 221
Figure 19. RDP level transition scheme when TrustZone is disabled (TZEN = 0) . . . 228
Figure 20. RDP level transition scheme when TrustZone is enabled (TZEN = 1) . . . 229
Figure 21. ICACHE block diagram . . . 279
Figure 22. ICACHE TAG and data memories functional view . . . 281
Figure 23. ICACHE remapping address mechanism . . . 284
Figure 24. Power supply overview . . . 296
Figure 25. Brownout reset waveform . . . 302
Figure 26. PVD thresholds . . . 303
Figure 27. I/O states in Stop 3 mode . . . 322
Figure 28. I/O states in Standby mode . . . 325
Figure 29. Simplified diagram of the reset circuit . . . 358
Figure 30. Clock tree . . . 361
Figure 31. MSI block diagram . . . 364
Figure 32. CRS block diagram . . . 444
Figure 33. CRS counter behavior . . . 446
Figure 34. Structure of three-volt or five-volt tolerant GPIO (TT or FT) . . . 455
Figure 35. Input floating/pull-up/pull-down configurations . . . 459
Figure 36. Output configuration . . . 460
Figure 37. Alternate function configuration . . . 461
Figure 38. High-impedance analog configuration . . . 461
Figure 39. I/O compensation cell block diagram . . . 477
Figure 40. GPDMA block diagram . . . 512
Figure 41. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 513
Figure 42. GPDMA channel suspend and resume sequence . . . 514
Figure 43. GPDMA channel abort and restart sequence . . . 515
Figure 44. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 516
Figure 45. Static linked-list data structure (all Uxx = 1)

of a 2D addressing channel x . . . 517
Figure 46. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 518
Figure 47. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 518
Figure 48. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 520
Figure 49. Inserting a LLIn with an auxiliary GPDMA channel y . . . 522
Figure 50. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 524
Figure 51. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 525
Figure 52. Replace with a new LLIn' in register file in link step mode . . . 526
Figure 53. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 527
Figure 54. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 528
Figure 55. GPDMA channel execution and linked-list programming . . . 530
Figure 56. Programmed 2D addressing . . . 533
Figure 57. GPDMA arbitration policy . . . 540
Figure 58. Trigger hit, memorization, and overrun waveform . . . 543
Figure 59. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 544
Figure 60. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 545
Figure 61. EXTI block diagram . . . 588
Figure 62. Configurable event trigger logic CPU wake-up . . . 590
Figure 63. EXTI mux GPIO selection . . . 591
Figure 64. CRC calculation unit block diagram . . . 604
Figure 65. OCTOSPI block diagram in octal configuration . . . 612
Figure 66. OCTOSPI block diagram in quad configuration . . . 612
Figure 67. OCTOSPI block diagram in dual-quad configuration . . . 613
Figure 68. SDR read command in octal configuration . . . 614
Figure 69. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 617
Figure 70. SDR write command in octo-SPI mode example . . . 619
Figure 71. DTR write in octal-SPI mode (Macronix mode) example . . . 620
Figure 72. Example of HyperBus read operation . . . 621
Figure 73. HyperBus write operation with initial latency . . . 623
Figure 74. HyperBus read operation with additional latency . . . 623
Figure 75. HyperBus write operation with additional latency . . . 624
Figure 76. HyperBus write operation with no latency (register write) . . . 624
Figure 77. HyperBus read operation page crossing with latency . . . 625
Figure 78. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 631
Figure 79. OctaRAM read operation with reverse data ordering D1/D0 . . . 632
Figure 80. NCS when CKMODE = 0 (T = CLK period) . . . 638
Figure 81. NCS when CKMODE = 1 in SDR mode (T = CLK period) . . . 638
Figure 82. NCS when CKMODE = 1 in DTR mode (T = CLK period) . . . 638
Figure 83. NCS when CKMODE = 1 with an abort (T = CLK period) . . . 639
Figure 84. SDMMC "no response" and "no data" operations . . . 668
Figure 85. SDMMC (multiple) block read operation . . . 668
Figure 86. SDMMC (multiple) block write operation . . . 669
Figure 87. SDMMC (sequential) stream read operation . . . 669
Figure 88. SDMMC (sequential) stream write operation . . . 669
Figure 89. SDMMC block diagram . . . 671
Figure 90. SDMMC Command and data phase relation . . . 673

Figure 91. Control unit . . . 675
Figure 92. Command/response path . . . 676
Figure 93. Command path state machine (CPSM) . . . 677
Figure 94. Data path . . . 683
Figure 95. DDR mode data packet clocking . . . 684
Figure 96. DDR mode CRC status / boot acknowledgment clocking . . . 684
Figure 97. Data path state machine (DPSM) . . . 685
Figure 98. CLKMUX unit . . . 696
Figure 99. Linked list structures . . . 698
Figure 100. Asynchronous interrupt generation . . . 701
Figure 101. Synchronous interrupt period data read . . . 702
Figure 102. Synchronous interrupt period data write . . . 702
Figure 103. Asynchronous interrupt period data read . . . 703
Figure 104. Asynchronous interrupt period data write . . . 704
Figure 105. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 707
Figure 106. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 707
Figure 107. Read Wait with SDMMC_CK < 50 MHz . . . 708
Figure 108. Read Wait with SDMMC_CK > 50 MHz . . . 708
Figure 109. CMD12 stream timing . . . 711
Figure 110. CMD5 Sleep Awake procedure . . . 713
Figure 111. Normal boot mode operation . . . 715
Figure 112. Alternative boot mode operation . . . 716
Figure 113. Command response R1b busy signaling . . . 717
Figure 114. SDMMC state control . . . 718
Figure 115. Card cycle power / power up diagram . . . 719
Figure 116. CMD11 signal voltage switch sequence . . . 720
Figure 117. Voltage switch transceiver typical application . . . 722
Figure 118. DLYB block diagram . . . 750
Figure 119. ADC block diagram . . . 758
Figure 120. ADC clock scheme . . . 762
Figure 121. ADC1 connectivity . . . 763
Figure 122. ADC2 connectivity . . . 764
Figure 123. ADC calibration . . . 766
Figure 124. Calibration factor forcing . . . 767
Figure 125. Enabling/disabling the ADC . . . 768
Figure 126. Bulb mode timing diagram . . . 771
Figure 127. Analog-to-digital conversion time . . . 775
Figure 128. Stopping ongoing regular conversions . . . 776
Figure 129. Stopping ongoing regular and injected conversions . . . 776
Figure 130. Triggers shared between ADC master and slave . . . 778
Figure 131. Injected conversion latency . . . 779
Figure 132. Example of ADC_JSQR queue of context (sequence change) . . . 783
Figure 133. Example of ADC_JSQR queue of context (trigger change) . . . 783
Figure 134. Example of ADC_JSQR queue of context with overflow before conversion . . . 784
Figure 135. Example of ADC_JSQR queue of context with overflow during conversion . . . 784
Figure 136. Example of ADC_JSQR queue of context with empty queue (case JQM = 0) . . . 785
Figure 137. Example of ADC_JSQR queue of context with empty queue (JQM = 1) . . . 786
Figure 138. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion . . . 786
Figure 139. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs . . . 787

Figure 140. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion . . . 787
Figure 141. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 788
Figure 142. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0) . . . 788
Figure 143. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1) . . . 789
Figure 144. Single conversions of a sequence, software trigger . . . 791
Figure 145. Continuous conversion of a sequence, software trigger . . . 791
Figure 146. Single conversions of a sequence, hardware trigger . . . 792
Figure 147. Continuous conversions of a sequence, hardware trigger . . . 792
Figure 148. Right alignment (offset disabled, unsigned value) . . . 794
Figure 149. Right alignment (offset enabled, signed value) . . . 795
Figure 150. Left alignment (offset disabled, unsigned value) . . . 795
Figure 151. Left alignment (offset enabled, signed value) . . . 796
Figure 152. Example of overrun (OVRMOD = 0) . . . 799
Figure 153. Example of overrun (OVRMOD = 1) . . . 799
Figure 154. AUTODLY = 1, regular conversion in continuous mode, software trigger . . . 803
Figure 155. AUTODLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 804
Figure 156. AUTODLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 805
Figure 157. AUTODLY = 1, regular continuous conversions interrupted by injected conversions . . . 805
Figure 158. AUTODLY = 1 in autoinjection mode (JAUTO = 1) . . . 806
Figure 159. Analog watchdog guarded area . . . 806
Figure 160. ADC_AWDy_OUT signal generation (on all regular channels) . . . 809
Figure 161. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 809
Figure 162. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 810
Figure 163. ADC_AWDy_OUT signal generation (on all injected channels) . . . 810
Figure 164. 12-bit result oversampling with 10-bit right shift and rounding . . . 811
Figure 165. Triggered regular oversampling mode (TROVS bit = 1) . . . 813
Figure 166. Regular oversampling modes (4x ratio) . . . 814
Figure 167. Regular and injected oversampling modes used simultaneously . . . 815
Figure 168. Triggered regular oversampling with injection . . . 815
Figure 169. Oversampling in autoinjection mode . . . 816
Figure 170. Dual ADC block diagram . . . 818
Figure 171. Injected simultaneous mode on four channels: Dual ADC mode . . . 819
Figure 172. Regular simultaneous mode on 16 channels: dual ADC mode . . . 821
Figure 173. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode . . . 823
Figure 174. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode . . . 823
Figure 175. Interleaved conversion with injection . . . 824
Figure 176. Alternate trigger: injected group of each ADC . . . 825
Figure 177. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 826
Figure 178. Alternate + regular simultaneous . . . 827
Figure 179. Case of trigger occurring during injected conversion . . . 827
Figure 180. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 828
Figure 181. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first . . . 828
Figure 182. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first . . . 828
Figure 183. DMA Requests in regular simultaneous mode when DAMDF[1:0]=0b00 . . . 829
Figure 184. DMA requests in interleaved mode when DAMDF = 0b10 . . . 830

Figure 185. Temperature sensor channel block diagram . . . 833
Figure 186. VBAT channel block diagram . . . 835
Figure 187. VREFINT channel block diagram . . . 835
Figure 188. Dual-channel DAC block diagram . . . 880
Figure 189. Data registers in single DAC channel mode . . . 883
Figure 190. Data registers in dual DAC channel mode . . . 884
Figure 191. Timing diagram for conversion with trigger disabled TEN = 0 . . . 885
Figure 192. DAC LFSR register calculation algorithm . . . 887
Figure 193. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 888
Figure 194. DAC triangle wave generation . . . 888
Figure 195. DAC conversion (SW trigger enabled) with triangle wave generation . . . 889
Figure 196. DAC sample and hold mode phase diagram . . . 891
Figure 197. VREFBUF block diagram . . . 919
Figure 198. Comparator block diagrams . . . 924
Figure 199. Window mode . . . 927
Figure 200. Comparator hysteresis . . . 927
Figure 201. Comparator output blanking . . . 928
Figure 202. Scaler . . . 929
Figure 203. Standalone mode: external gain setting mode . . . 936
Figure 204. Follower configuration . . . 937
Figure 205. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used . . . 938
Figure 206. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering . . . 939
Figure 207. ADF block diagram . . . 949
Figure 208. SITF overview . . . 951
Figure 209. SPI timing example . . . 952
Figure 210. Manchester timing example (SITFMOD = 11) . . . 954
Figure 211. CKGEN overview . . . 956
Figure 212. BSMX overview . . . 958
Figure 213. DFLT overview . . . 959
Figure 214. Programmable delay . . . 960
Figure 215. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16 . . . 962
Figure 216. Reshape filter frequency response normalized (FRS / 2 = 1) . . . 967
Figure 217. Trigger logic for DFLT and CKGEN . . . 969
Figure 218. Asynchronous continuous mode (ACQMOD[2:0] = 0) . . . 970
Figure 219. Asynchronous single-shot mode (ACQMOD[2:0] = 001) . . . 971
Figure 220. Synchronous continuous mode (ACQMOD[2:0] = 010) . . . 972
Figure 221. Synchronous single-shot mode (ACQMOD[2:0] = 011) . . . 973
Figure 222. Window continuous mode (ACQMOD[2:0] = 100) . . . 974
Figure 223. Discard function example . . . 976
Figure 224. Start sequence with DFLTEN, in continuous mode, audio configuration . . . 976
Figure 225. SAD block diagram . . . 977
Figure 226. SAD flow diagram . . . 979
Figure 227. SAD timing diagram example . . . 985
Figure 228. ADF_DFLTxDR data format . . . 985
Figure 229. Data re-synchronization . . . 986
Figure 230. Example of overflow and transfer to memory . . . 987
Figure 231. ADF interrupt interface . . . 990
Figure 232. Sensor connection examples . . . 995
Figure 233. Global frequency response . . . 995
Figure 234. Detailed frequency response . . . 996
Figure 235. Simplified DFLT view with gain information . . . 998

Figure 236. SAD example working with SADMOD = 01 . . . 1001
Figure 237. SAD example working with SADMOD = 1x . . . 1003
Figure 238. TSC block diagram . . . 1024
Figure 239. Surface charge transfer analog I/O group structure . . . 1025
Figure 240. Sampling capacitor voltage variation . . . 1026
Figure 241. Charge transfer acquisition sequence . . . 1027
Figure 242. Spread spectrum variation principle . . . 1028
Figure 243. CCB block diagram . . . 1042
Figure 244. CCB operation CPU → PKA + SAES (write) . . . 1044
Figure 245. CCB operation PKA → SAES . . . 1044
Figure 246. CCB operation SAES → PKA . . . 1044
Figure 247. CCB operation SAES → CCB . . . 1045
Figure 248. CCB operation RNG → PKA (write) . . . 1045
Figure 249. CCB operation RNG → SAES (write) . . . 1046
Figure 250. Blob creation scenarios . . . 1047
Figure 251. RNG block diagram . . . 1066
Figure 252. NIST SP800-90B entropy source model . . . 1067
Figure 253. RNG initialization overview . . . 1070
Figure 254. AES block diagram . . . 1083
Figure 255. Encryption/ decryption typical usage . . . 1085
Figure 256. Typical operation with authentication . . . 1087
Figure 257. Example of suspend mode management . . . 1088
Figure 258. ECB encryption . . . 1089
Figure 259. ECB decryption . . . 1089
Figure 260. CBC encryption . . . 1090
Figure 261. CBC decryption . . . 1090
Figure 262. Message construction in CTR mode . . . 1093
Figure 263. CTR encryption . . . 1093
Figure 264. Message construction in GCM . . . 1095
Figure 265. GCM authenticated encryption . . . 1096
Figure 266. Message construction in GMAC mode . . . 1099
Figure 267. GMAC authentication mode . . . 1100
Figure 268. Message construction in CCM mode . . . 1101
Figure 269. CCM mode authenticated encryption . . . 1102
Figure 270. 128-bit block construction according to the data type . . . 1107
Figure 271. SAES block diagram . . . 1129
Figure 272. Encryption/ decryption typical usage . . . 1131
Figure 273. Typical operation with authentication . . . 1133
Figure 274. Example of suspend mode management . . . 1134
Figure 275. ECB encryption . . . 1135
Figure 276. ECB decryption . . . 1135
Figure 277. CBC encryption . . . 1136
Figure 278. CBC decryption . . . 1136
Figure 279. Message construction in CTR mode . . . 1139
Figure 280. CTR encryption . . . 1140
Figure 281. Message construction in GCM . . . 1141
Figure 282. GCM authenticated encryption . . . 1143
Figure 283. Message construction in GMAC mode . . . 1146
Figure 284. GMAC authentication mode . . . 1146
Figure 285. Message construction in CCM mode . . . 1147
Figure 286. CCM mode authenticated encryption . . . 1149
Figure 287. Operation with wrapped keys for SAES in ECB and CBC modes . . . 1152

Figure 288. Operation with wrapped keys for SAES in CTR mode 1155
Figure 289. Usage of Shared-key mode 1156
Figure 290. 128-bit block construction according to the data type 1159
Figure 291. Key protection mechanisms 1161
Figure 292. HASH block diagram 1183
Figure 293. Message data swapping feature 1185
Figure 294. HASH suspend/resume mechanism 1191
Figure 295. PKA block diagram 1205
Figure 296. Chaining of PKA with SAES or RNG 1212
Figure 297. Advanced-control timer block diagram 1244
Figure 298. Counter timing diagram with prescaler division change from 1 to 2 1250
Figure 299. Counter timing diagram with prescaler division change from 1 to 4 1251
Figure 300. Counter timing diagram, internal clock divided by 1 1252
Figure 301. Counter timing diagram, internal clock divided by 2 1253
Figure 302. Counter timing diagram, internal clock divided by 4 1253
Figure 303. Counter timing diagram, internal clock divided by N 1254
Figure 304. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) 1254
Figure 305. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) 1255
Figure 306. Counter timing diagram, internal clock divided by 1 1256
Figure 307. Counter timing diagram, internal clock divided by 2 1257
Figure 308. Counter timing diagram, internal clock divided by 4 1257
Figure 309. Counter timing diagram, internal clock divided by N 1258
Figure 310. Counter timing diagram, update event when repetition counter is not used 1258
Figure 311. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 1260
Figure 312. Counter timing diagram, internal clock divided by 2 1260
Figure 313. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 1261
Figure 314. Counter timing diagram, internal clock divided by N 1261
Figure 315. Counter timing diagram, update event with ARPE = 1 (counter underflow) 1262
Figure 316. Counter timing diagram, Update event with ARPE = 1 (counter overflow) 1263
Figure 317. Update rate examples depending on mode and TIMx_RCR register settings 1264
Figure 319. Control circuit in normal mode, internal clock divided by 1 1266
Figure 320. tim_ti2 external clock connection example 1266
Figure 321. Control circuit in external clock mode 1 1267
Figure 322. External trigger input block 1268
Figure 323. Control circuit in external clock mode 2 1269
Figure 324. Capture/compare channel (example: channel 1 input stage) 1269
Figure 325. Capture/compare channel 1 main circuit 1270
Figure 326. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) 1271
Figure 327. Output stage of capture/compare channel (channel 5, idem ch. 6) 1271
Figure 328. PWM input mode timing 1274
Figure 329. Output compare mode, toggle on tim_oc1 1276
Figure 330. Edge-aligned PWM waveforms (ARR = 8) 1277
Figure 331. Center-aligned PWM waveforms (ARR = 8) 1278
Figure 332. Dithering principle 1279
Figure 333. Data format and register coding in dithering mode 1280
Figure 334. PWM resolution vs frequency 1281
Figure 335. PWM dithering pattern 1282
Figure 336. Dithering effect on duty cycle in center-aligned PWM mode 1283
Figure 337. Generation of 2 phase-shifted PWM signals with 50% duty cycle 1285
Figure 338. Combined PWM mode on channel 1 and 3 1286

Figure 339. 3-phase combined PWM signals with multiple trigger pulses per period 1287
Figure 340. Complementary output with symmetrical dead-time insertion 1288
Figure 341. Asymmetrical deadtime 1289
Figure 342. Dead-time waveforms with delay greater than the negative pulse 1289
Figure 343. Dead-time waveforms with delay greater than the positive pulse. 1289
Figure 344. Break and Break2 circuitry overview 1292
Figure 345. Various output behavior in response to a break event on tim_brk (OSSI = 1) 1294
Figure 346. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) 1295
Figure 347. PWM output state following tim_brk assertion (OSSI = 0) 1296
Figure 348. Output redirection (tim_brk2 request not represented). 1297
Figure 349. tim_ocref_clr input selection multiplexer. 1298
Figure 350. Clearing TIMx tim_ocxref 1299
Figure 351. 6-step generation, COM example (OSSR = 1). 1300
Figure 352. Example of one pulse mode. 1301
Figure 353. Retriggerable one-pulse mode 1302
Figure 354. Pulse generator circuitry 1303
Figure 355. Pulse generation on compare event, for edge-aligned and encoder modes 1304
Figure 356. Extended pulsewidth in case of concurrent triggers 1305
Figure 357. Example of counter operation in encoder interface mode. 1307
Figure 358. Example of encoder interface mode with tim_ti1fp1 polarity inverted. 1307
Figure 359. Quadrature encoder counting modes 1308
Figure 360. Direction plus clock encoder mode. 1309
Figure 361. Directional clock encoder mode (CC1P = CC2P = 0). 1309
Figure 362. Directional clock encoder mode (CC1P = CC2P = 1). 1310
Figure 363. Index gating options 1311
Figure 364. Jittered Index signals 1311
Figure 365. Index generation for IPOS[1:0] = 11 1312
Figure 366. Counter reading with index gated on channel A (IPOS[1:0] = 11) 1312
Figure 367. Counter reading with index ungated (IPOS[1:0] = 00) 1313
Figure 368. Counter reading with index gated on channel A and B. 1313
Figure 369. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11). 1314
Figure 370. Counter reset Narrow index pulse (closer view, ARR = 0x07) 1315
Figure 371. Index behavior in x1 and x2 mode (IPOS[1:0] = 01). 1316
Figure 372. Directional index sensitivity. 1316
Figure 373. Counter reset as function of FIDX bit setting 1317
Figure 374. Index blanking. 1317
Figure 375. Index behavior in clock + direction mode, IPOS[0] = 1. 1318
Figure 376. Index behavior in directional clock mode, IPOS[0] = 1 1318
Figure 377. State diagram for quadrature encoded signals. 1319
Figure 378. Up-counting encoder error detection 1320
Figure 379. Down-counting encode error detection. 1321
Figure 380. Encoder mode change with preload transferred on update (SMSPS = 0) 1322
Figure 381. Measuring time interval between edges on three signals. 1323
Figure 382. Example of Hall sensor interface 1325
Figure 383. Control circuit in reset mode. 1326
Figure 384. Control circuit in Gated mode 1327
Figure 385. Control circuit in trigger mode. 1328
Figure 386. Control circuit in external clock mode 2 + trigger mode 1329
Figure 387. General-purpose timer block diagram 1388
Figure 388. Counter timing diagram with prescaler division change from 1 to 2 1393
Figure 389. Counter timing diagram with prescaler division change from 1 to 4 1394
Figure 390. Counter timing diagram, internal clock divided by 1 1395

Figure 391. Counter timing diagram, internal clock divided by 2 1395
Figure 392. Counter timing diagram, internal clock divided by 4 1396
Figure 393. Counter timing diagram, internal clock divided by N. 1396
Figure 394. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded). . . 1397
Figure 395. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded). 1398
Figure 396. Counter timing diagram, internal clock divided by 1 1399
Figure 397. Counter timing diagram, internal clock divided by 2 1400
Figure 398. Counter timing diagram, internal clock divided by 4 1400
Figure 399. Counter timing diagram, internal clock divided by N. 1401
Figure 400. Counter timing diagram, Update event 1401
Figure 401. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 1403
Figure 402. Counter timing diagram, internal clock divided by 2 1403
Figure 403. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 1404
Figure 404. Counter timing diagram, internal clock divided by N. 1404
Figure 405. Counter timing diagram, Update event with ARPE = 1 (counter underflow). 1405
Figure 406. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 1406
Figure 407. Control circuit in normal mode, internal clock divided by 1 1407
Figure 408. tim_ti2 external clock connection example 1407
Figure 409. Control circuit in external clock mode 1 1408
Figure 410. External trigger input block 1409
Figure 411. Control circuit in external clock mode 2 1410
Figure 412. Capture/compare channel (example: channel 1 input stage) 1410
Figure 413. Capture/compare channel 1 main circuit 1411
Figure 414. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4) 1411
Figure 415. PWM input mode timing 1414
Figure 416. Output compare mode, toggle on tim_oc1 1416
Figure 417. Edge-aligned PWM waveforms (ARR = 8) 1417
Figure 418. Center-aligned PWM waveforms (ARR = 8). 1418
Figure 419. Dithering principle 1419
Figure 420. Data format and register coding in dithering mode 1420
Figure 421. PWM resolution vs frequency (16-bit mode). 1421
Figure 422. PWM resolution vs frequency (32-bit mode). 1421
Figure 423. PWM dithering pattern 1422
Figure 424. Dithering effect on duty cycle in center-aligned PWM mode 1423
Figure 425. Generation of two phase-shifted PWM signals with 50% duty cycle 1425
Figure 426. Combined PWM mode on channels 1 and 3 1426
Figure 427. OCREF_CLR input selection multiplexer 1427
Figure 428. Clearing TIMx tim_ocxref 1427
Figure 429. Example of One-pulse mode 1428
Figure 430. Retriggerable one-pulse mode 1430
Figure 431. Pulse generator circuitry 1431
Figure 432. Pulse generation on compare event, for edge-aligned and encoder modes 1431
Figure 433. Extended pulse width in case of concurrent triggers 1432
Figure 434. Example of counter operation in encoder interface mode 1434
Figure 435. Example of encoder interface mode with tim_ti1fp1 polarity inverted 1434
Figure 436. Quadrature encoder counting modes 1435
Figure 437. Direction plus clock encoder mode. 1436
Figure 438. Directional clock encoder mode (CC1P = CC2P = 0). 1437
Figure 439. Directional clock encoder mode (CC1P = CC2P = 1). 1437
Figure 440. Index gating options 1439
Figure 441. Jittered Index signals 1439
Figure 442. Index generation for IPOS[1:0] = 11 1440

Figure 443. Counter reading with index gated on channel A (IPOS[1:0] = 11) 1440
Figure 444. Counter reading with index ungated (IPOS[1:0] = 00) 1441
Figure 445. Counter reading with index gated on channel A and B 1441
Figure 446. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) 1442
Figure 447. Counter reset Narrow index pulse (closer view, ARR = 0x07) 1443
Figure 448. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) 1444
Figure 449. Directional index sensitivity 1444
Figure 450. Counter reset as function of FIDX bit setting 1445
Figure 451. Index blanking 1445
Figure 452. Index behavior in clock + direction mode, IPOS[0] = 1 1446
Figure 453. Index behavior in directional clock mode, IPOS[0] = 1 1446
Figure 454. State diagram for quadrature encoded signals 1447
Figure 455. Up-counting encoder error detection 1448
Figure 456. Down-counting encode error detection 1449
Figure 457. Encoder mode change with preload transferred on update (SMSPS = 0) 1450
Figure 458. Control circuit in reset mode 1452
Figure 459. Control circuit in gated mode 1453
Figure 460. Control circuit in trigger mode 1453
Figure 461. Control circuit in external clock mode 2 + trigger mode 1455
Figure 462. Master/Slave timer example 1455
Figure 463. Master/slave connection example with 1 channel only timers 1456
Figure 464. Gating TIM_slv with tim_oc1ref of TIM_mstr 1457
Figure 465. Gating TIM_slv with Enable of TIM_mstr 1458
Figure 466. Triggering TIM_slv with update of TIM_mstr 1459
Figure 467. Triggering TIM_slv with Enable of TIM_mstr 1459
Figure 468. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input 1460
Figure 469. Basic timer block diagram 1502
Figure 470. Control circuit in normal mode, internal clock divided by 1 1503
Figure 471. Counter timing diagram with prescaler division change from 1 to 2 1504
Figure 472. Counter timing diagram with prescaler division change from 1 to 4 1505
Figure 473. Counter timing diagram, internal clock divided by 1 1506
Figure 474. Counter timing diagram, internal clock divided by 2 1506
Figure 475. Counter timing diagram, internal clock divided by 4 1507
Figure 476. Counter timing diagram, internal clock divided by N 1507
Figure 477. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) 1508
Figure 478. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) 1509
Figure 479. Dithering principle 1510
Figure 480. Data format and register coding in dithering mode 1510
Figure 481. FCnt resolution vs frequency 1511
Figure 482. PWM dithering pattern 1511
Figure 483. TIM15 block diagram 1524
Figure 484. TIM16/TIM17 block diagram 1525
Figure 485. Counter timing diagram with prescaler division change from 1 to 2 1529
Figure 486. Counter timing diagram with prescaler division change from 1 to 4 1530
Figure 487. Counter timing diagram, internal clock divided by 1 1531
Figure 488. Counter timing diagram, internal clock divided by 2 1532
Figure 489. Counter timing diagram, internal clock divided by 4 1532
Figure 490. Counter timing diagram, internal clock divided by N 1533
Figure 491. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) 1533

Figure 492. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 1534
Figure 493. Update rate examples depending on mode and TIMx_RCR register settings . . . 1535
Figure 494. Control circuit in normal mode, internal clock divided by 1 . . . 1536
Figure 495. tim_ti2 external clock connection example . . . 1536
Figure 496. Control circuit in external clock mode 1 . . . 1537
Figure 497. Capture/compare channel (example: channel 1 input stage) . . . 1538
Figure 498. Capture/compare channel 1 main circuit . . . 1538
Figure 499. Output stage of capture/compare channel (channel 1) . . . 1539
Figure 500. Output stage of capture/compare channel (channel 2 for TIM15) . . . 1539
Figure 501. PWM input mode timing . . . 1542
Figure 502. Output compare mode, toggle on tim_oc1 . . . 1544
Figure 503. Edge-aligned PWM waveforms (ARR = 8) . . . 1545
Figure 504. Dithering principle . . . 1546
Figure 505. Data format and register coding in dithering mode . . . 1546
Figure 506. PWM resolution vs frequency . . . 1547
Figure 507. PWM dithering pattern . . . 1548
Figure 508. Combined PWM mode on channel 1 and 2 . . . 1550
Figure 509. Complementary output with symmetrical dead-time insertion . . . 1551
Figure 510. Asymmetrical deadtime . . . 1552
Figure 511. Dead-time waveforms with delay greater than the negative pulse . . . 1552
Figure 512. Dead-time waveforms with delay greater than the positive pulse . . . 1552
Figure 513. Break circuitry overview . . . 1554
Figure 514. Output behavior in response to a break event on tim_brk . . . 1556
Figure 515. Output redirection . . . 1558
Figure 516. tim_ocref_clr input selection multiplexer . . . 1559
Figure 517. 6-step generation, COM example (OSSR = 1) . . . 1560
Figure 518. Example of one pulse mode . . . 1561
Figure 519. Retriggerable one pulse mode . . . 1563
Figure 520. Measuring time interval between edges on 2 signals . . . 1563
Figure 521. Control circuit in reset mode . . . 1564
Figure 522. Control circuit in gated mode . . . 1565
Figure 523. Control circuit in trigger mode . . . 1566
Figure 524. LPTIM1/2/3 block diagram(1) . . . 1632
Figure 525. LPTIM4 block diagram(1) . . . 1633
Figure 526. Glitch filter timing diagram . . . 1637
Figure 527. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 1639
Figure 528. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1640
Figure 529. LPTIM output waveform, Continuous counting mode configuration . . . 1640
Figure 530. Waveform generation . . . 1642
Figure 531. Encoder mode counting sequence . . . 1646
Figure 532. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) . . . 1647
Figure 533. Capture/compare input stage (channel 1) . . . 1648
Figure 534. Capture/compare output stage (channel 1) . . . 1648
Figure 535. Edge-aligned PWM mode (PRELOAD = 1) . . . 1650
Figure 536. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 1651
Figure 537. PWM mode with immediate update versus preloaded update . . . 1652
Figure 538. IRTIM internal hardware connections . . . 1682
Figure 539. Independent watchdog block diagram . . . 1685

Figure 540. Reset timing due to timeout . . . 1687
Figure 541. Reset timing due to refresh in the not allowed area . . . 1688
Figure 542. Changing PR, RL, and performing a refresh(1) . . . 1689
Figure 543. Independent watchdog interrupt timing diagram . . . 1691
Figure 544. Watchdog block diagram . . . 1699
Figure 545. Window watchdog timing diagram . . . 1701
Figure 546. RTC block diagram . . . 1707
Figure 547. TAMP block diagram . . . 1763
Figure 548. Backup registers protection zones . . . 1768
Figure 549. Tamper sampling with precharge pulse . . . 1774
Figure 550. Low level detection with precharge and filtering . . . 1774
Figure 551. Block diagram . . . 1794
Figure 552. I²C-bus protocol . . . 1796
Figure 553. Setup and hold timings . . . 1798
Figure 554. I2C initialization flow . . . 1801
Figure 555. Data reception . . . 1802
Figure 556. Data transmission . . . 1802
Figure 557. Target initialization flow . . . 1806
Figure 558. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 1807
Figure 559. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 1808
Figure 560. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 1809
Figure 561. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 1810
Figure 562. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 1811
Figure 563. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 1811
Figure 564. Controller clock generation . . . 1813
Figure 565. Controller initialization flow . . . 1815
Figure 566. 10-bit address read access with HEAD10R = 0 . . . 1815
Figure 567. 10-bit address read access with HEAD10R = 1 . . . 1816
Figure 568. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 1817
Figure 569. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 1818
Figure 570. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 1819
Figure 571. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 1821
Figure 572. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 1822
Figure 573. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 1823
Figure 574. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 1827
Figure 575. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 1831
Figure 576. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 1831
Figure 577. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 1833
Figure 578. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 1834
Figure 579. Bus transfer diagrams for SMBus controller transmitter . . . 1835
Figure 580. Bus transfer diagrams for SMBus controller receiver . . . 1837
Figure 581. I3C block diagram . . . 1865
Figure 582. I3C (primary) controller state and programming sequence diagram . . . 1869
Figure 583. I3C target state and programing sequence diagram . . . 1874
Figure 584. I3C CCC messages, as controller . . . 1888
Figure 585. I3C broadcast ENTDAA CCC, as controller . . . 1889
Figure 586. I3C broadcast, direct read and direct write RSTACT CCC, as controller . . . 1890
Figure 587. I3C CCC messages, as target . . . 1892
Figure 588. I3C broadcast ENTDAA CCC, as target . . . 1893

Figure 589. I3C broadcast DEFTGTS CCC, as target . . . 1894
Figure 590. I3C broadcast DEFGRPA CCC, as target . . . 1895
Figure 591. I3C private read/write messages, as controller . . . 1897
Figure 592. I3C private read/write messages, as target . . . 1898
Figure 593. Legacy I2C read/write messages, as controller . . . 1899
Figure 594. IBI transfer, as controller/target . . . 1900
Figure 595. Hot-join request transfer, as controller/target . . . 1902
Figure 596. Controller-role request transfer, as controller/target . . . 1903
Figure 597. C-FIFO management, as controller . . . 1904
Figure 598. TX-FIFO management, as controller . . . 1906
Figure 599. RX-FIFO management, as controller . . . 1908
Figure 600. S-FIFO management, as controller . . . 1911
Figure 601. RX-FIFO management, as target . . . 1912
Figure 602. TX-FIFO management with I3C_TGTTDR, as target . . . 1914
Figure 603. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target . . . 1916
Figure 604. USART block diagram . . . 1976
Figure 605. Word length programming . . . 1980
Figure 606. Configurable stop bits . . . 1982
Figure 607. TC/TXE behavior when transmitting . . . 1984
Figure 608. Start bit detection when oversampling by 16 or 8 . . . 1985
Figure 609. usart_ker_ck clock divider block diagram . . . 1988
Figure 610. Data sampling when oversampling by 16 . . . 1989
Figure 611. Data sampling when oversampling by 8 . . . 1990
Figure 612. Mute mode using Idle line detection . . . 1997
Figure 613. Mute mode using address mark detection . . . 1998
Figure 614. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 2001
Figure 615. Break detection in LIN mode vs. Framing error detection . . . 2002
Figure 616. USART example of synchronous master transmission . . . 2003
Figure 617. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 2003
Figure 618. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . 2004
Figure 619. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 2005
Figure 620. ISO 7816-3 asynchronous protocol . . . 2007
Figure 621. Parity error detection using the 1.5 stop bits . . . 2009
Figure 622. IrDA SIR ENDEC block diagram . . . 2013
Figure 623. IrDA data modulation (3/16) - normal mode . . . 2013
Figure 624. Transmission using DMA . . . 2015
Figure 625. Reception using DMA . . . 2016
Figure 626. Hardware flow control between two USARTs . . . 2016
Figure 627. RS232 RTS flow control . . . 2017
Figure 628. RS232 CTS flow control . . . 2018
Figure 629. LPUART block diagram . . . 2064
Figure 630. LPUART word length programming . . . 2068
Figure 631. Configurable stop bits . . . 2070
Figure 632. TC/TXE behavior when transmitting . . . 2072
Figure 633. lpuart_ker_ck clock divider block diagram . . . 2076
Figure 634. Mute mode using Idle line detection . . . 2080
Figure 635. Mute mode using address mark detection . . . 2081
Figure 636. Transmission using DMA . . . 2083

Figure 637. Reception using DMA . . . 2084
Figure 638. Hardware flow control between two LPUARTs . . . 2085
Figure 639. RS232 RTS flow control . . . 2085
Figure 640. RS232 CTS flow control . . . 2086
Figure 641. SPI block diagram . . . 2120
Figure 642. Full-duplex single master/ single slave application . . . 2124
Figure 643. Half-duplex single master/ single slave application . . . 2125
Figure 644. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 2126
Figure 645. Master and three independent slaves connected in star topology . . . 2127
Figure 646. Multimaster application . . . 2128
Figure 647. Scheme of NSS control logic . . . 2130
Figure 648. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 2130
Figure 649. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 2131
Figure 650. Data clock timing diagram . . . 2133
Figure 651. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 2134
Figure 652. TI mode transfer . . . 2144
Figure 653. Optional configurations of the slave behavior when an underrun condition is detected . 2146
Figure 654. SAI functional block diagram . . . 2170
Figure 655. Audio frame . . . 2173
Figure 656. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 2175
Figure 657. FS role is start of frame (FSDEF = 0) . . . 2176
Figure 658. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 2177
Figure 659. First bit offset . . . 2177
Figure 660. Audio block clock generator overview . . . 2179
Figure 661. PDM typical connection and timing . . . 2183
Figure 662. Detailed PDM interface block diagram . . . 2184
Figure 663. Start-up sequence . . . 2185
Figure 664. SAI_ADR format in TDM mode, 32-bit slot width . . . 2186
Figure 665. SAI_ADR format in TDM mode, 16-bit slot width . . . 2187
Figure 666. SAI_ADR format in TDM mode, 8-bit slot width . . . 2188
Figure 667. AC’97 audio frame . . . 2191
Figure 668. SPDIF format . . . 2192
Figure 669. SAI_xDR register ordering . . . 2193
Figure 670. Data companding hardware in an audio block in the SAI . . . 2196
Figure 671. Tristate strategy on SD output line on an inactive slot . . . 2198
Figure 672. Tristate on output data line in a protocol like I2S . . . 2199
Figure 673. Overrun detection error . . . 2200
Figure 674. FIFO underrun event . . . 2200
Figure 675. CAN subsystem . . . 2234
Figure 676. FDCAN block diagram . . . 2236
Figure 677. Bit timing . . . 2238
Figure 678. Transceiver delay measurement . . . 2243
Figure 679. Pin control in bus monitoring mode . . . 2244
Figure 680. Pin control in loop-back mode . . . 2247
Figure 681. CAN error state diagram . . . 2248
Figure 682. Message RAM configuration . . . 2249
Figure 683. Standard message ID filter path . . . 2252
Figure 684. Extended message ID filter path . . . 2253
Figure 685. USB peripheral block diagram . . . 2300
Figure 686. Packet buffer areas with examples of buffer description table locations . . . 2307
Figure 687. Block diagram of debug support infrastructure . . . 2348

Figure 688. JTAG TAP state machine 2352

1 Documentation conventions

1.1 General information

The STM32U3 series devices have an Arm®(a) Cortex®-M33 core, with Arm® TrustZone®.

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.

1.5 Availability of peripherals

For detailed availability of peripherals and their number across all sales types (depending on product line and package), refer to the device datasheet.

2 Memory and bus architecture

2.1 System architecture

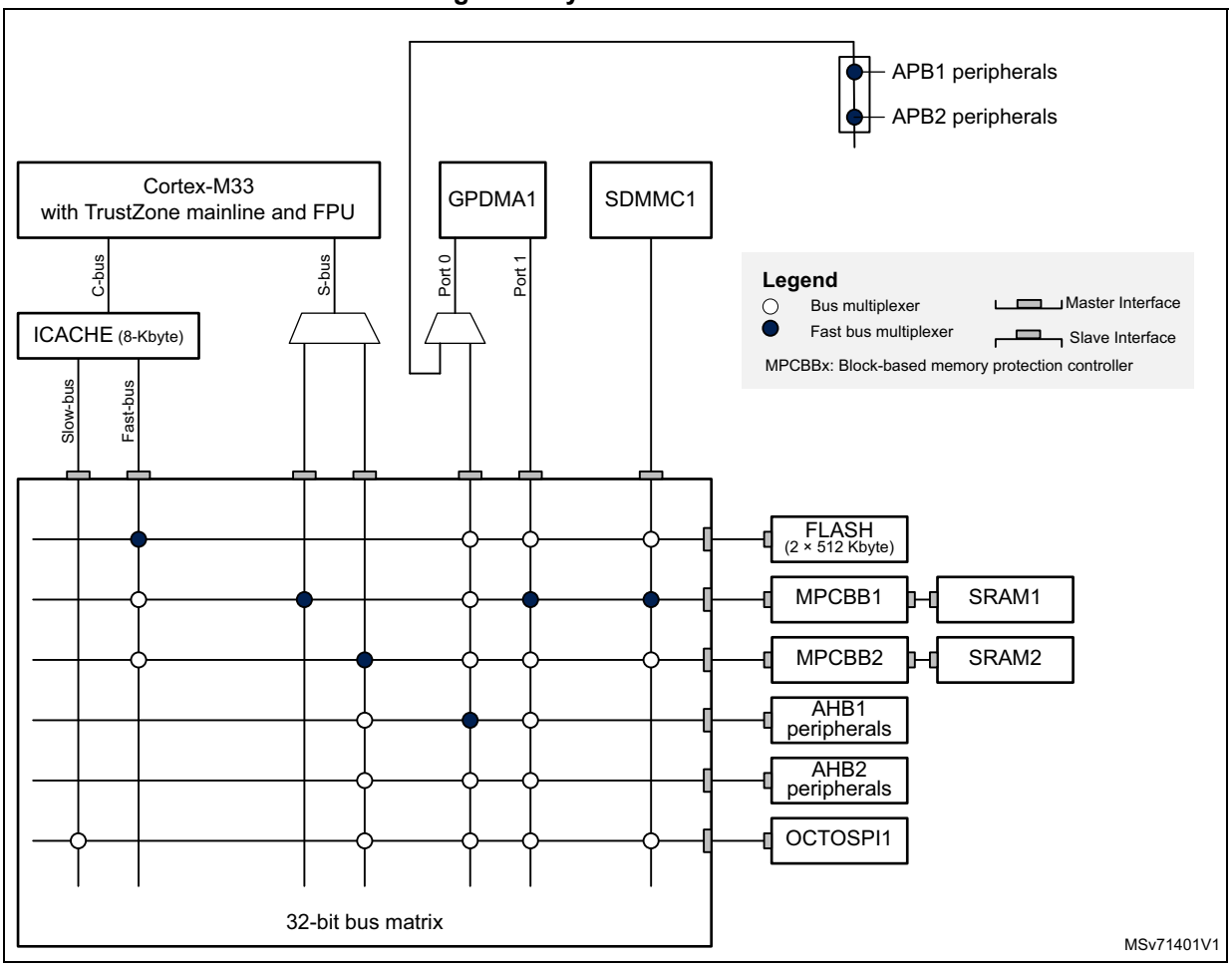
The STM32U3 series architecture relies on an Arm Cortex-M33 core optimized for execution thanks to an instruction cache having a direct access to the embedded flash memory.

This architecture also features a 32-bit multilayer AHB bus matrix that interconnects:

- up to 7 masters:
  - Fast C-bus, connecting Cortex-M33 with Arm TrustZone mainline and FPU core to the internal SRAMs and flash memory through the instruction cache
  - Slow C-bus, connecting Cortex-M33 with Arm TrustZone mainline and FPU core to the OCTOSPI1 external memory through the instruction cache
  - Cortex-M33 with Arm TrustZone mainline and FPU core S-bus (two masters connected to two internal SRAMs without latency), and to peripherals and OCTOSPI1 external memory
  - GPDMA1 (general purpose DMA featuring two master ports)
  - SDMMC1
- up to 6 slaves:
  - internal flash memory (1 Mbytes)
  - internal SRAM1 (192 Kbytes)
  - internal SRAM2 (64 Kbytes)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - OCTOSPI1

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in the figure below.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the internal flash memory and to the bus matrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in the code region. This bus targets the internal flash memory and the internal SRAMs (SRAM1 and SRAM2).

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex-M33 core to the bus matrix via the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in the code region. This bus targets the external memories (OCTOSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex-M33 core to the bus matrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAMs (SRAM1 and SRAM2), the AHB1 peripherals including the APB1 and APB2 peripherals, and the AHB2 peripherals.

SRAM1 and SRAM2 are accessible on this bus with a continuous mapping.

Note: The bus matrix has a zero latency when accessing SRAM1 and SRAM2*.*

2.1.4 GPDMA-bus

These buses connect the two AHB master interfaces of the GPDMA to the bus matrix. These buses target the internal flash memory, the internal SRAMs (SRAM1 and SRAM2), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the APB3 peripherals and the external memories through OCTOSPI.

2.1.5 SDMMC1 controller DMA bus

This bus connects the SDMMC1 DMA master interface to the bus matrix. This bus is used only by the SDMMC1 DMA to load/store data from/to the memory. This bus targets the data memories: internal flash memory, internal SRAMs (SRAM1 and SRAM2) and external memories through OCTOSPI.

2.1.6 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm. This bus matrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1*). For the same master, other slaves undergo a latency of at least one cycle at each new access.

2.1.7 AHB/APB bridges

The three AHB/APB bridges provide full synchronous connections between the AHB and the APB buses, allowing flexible selection of the peripheral frequency.

Refer to Memory and bus architecture for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for the internal SRAMs and flash memory interfaces). Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

Note: When an 8- or 16-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 8- or 16-bit data to feed the 32-bit vector.

2.2 Arm TrustZone security architecture

The security architecture is based on Arm TrustZone with the Armv8-M mainline extension.

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

When the TrustZone is enabled, the SAU (security-attribution unit) and IDAU (implementation-defined-attribution unit) defines the access permissions based on secure and nonsecure states.

- SAU: Up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as nonsecure or nonsecure callable attributes. The IDAU memory map partition is not configurable and fixed by hardware implementation (refer to *Figure 1* and to *Table 4* in *Section 2.3.2: Memory map and*

register boundary addresses). It is then combined with the results from the SAU security attribution and the higher security state is selected.

Based on IDAU security attribution, the flash memory, system SRAMs, and peripherals memory space are aliased twice for secure and nonsecure states. The external memories space is not aliased.

The table below shows an example of typical eight SAU regions mapping based on IDAU regions. The user can split and choose the secure, nonsecure or NSC regions for external memories as needed.

Table 1. Example of memory map security attribution versus SAU configuration regions

Region description	Address range	IDAU security attribution	SAU security attribution typical configuration	Final security attribution
Code - external memories	0x0000_0000 0x07FF_FFFF	Nonsecure	Secure, nonsecure or NSC(1)	Secure, nonsecure, or NSC
Code - flash memory and SRAM	0x0800_0000 0x0BFF_FFFF	Nonsecure	Nonsecure	Nonsecure
	0x0C00_0000 0x0FFF_FFFF	NSC	Secure or NSC	Secure or NSC
Code - external memories	0x1000_0000 0x17FF_FFFF	Nonsecure	Nonsecure	
	0x1800_0000 0x1FFF_FFFF			
SRAM	0x2000_0000 0x2FFF_FFFFF	Nonsecure		
	0x3000_0000 0x3FFF_FFFFF	NSC	Secure or NSC	Secure or NSC
Peripherals	0x4000_0000 0x4FFF_FFFFF	Nonsecure	Nonsecure	Nonsecure
	0x5000_0000 0x5FFF_FFFFF	NSC	Secure or NSC	Secure or NSC
External memories	0x6000_0000 0xDFFF_FFFF	Nonsecure	Secure, nonsecure or NSC	Secure, nonsecure or NSC

1. NSC = nonsecure callable

2.2.1 Default Arm TrustZone security state

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR, the default system security state is detailed below:

- CPU:
  - Cortex-M33 is in secure state after reset. The boot address must be at a secure address.
- Memory map:
  - SAU is fully secure after reset. Consequently, all memory map is fully secure. Up to height SAU configurable regions are available for security attribution.
- Flash memory:
  - The flash memory security area is defined by watermark user options.
  - Flash block-based security attributions are nonsecure after reset.
- SRAMs:
  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.

- External memories:
  - OCTOSPI does not support any TrustZone protection.
- Peripherals (see *Table 2* and *Table 3* for a list of securable and TrustZone-aware peripherals)
  - Securable peripherals are nonsecure after reset.
  - TrustZone-aware peripherals are nonsecure after reset. Their secure configuration registers are secure.
- All GPIOs secure after reset
- Interrupts:
  - NVIC: All interrupts are secure after reset. NVIC is banked for secure and nonsecure state.
  - TZIC: All illegal access interrupts are disabled after reset (see *Section 2.2: Arm TrustZone security architecture*).

2.2.2 Arm TrustZone peripheral classification

When the TrustZone security is active, a peripheral can be either securable or TrustZone-aware type as follows:

- Securable: peripheral protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties
- TrustZone-aware: peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure.

Refer to *Section 2.2: Arm TrustZone security architecture* for more details.

The tables below list the securable and TrustZone-aware peripherals within the system.

Table 2. Securable peripherals by TZSC

Bus	Peripheral
AHB2	DAC1
	OCTOSPI1
	SDMMC1
	CCB
	SAES
	PKA (+RAM)
	RNG
	HASH
	AES
	ADC2
	ADC1

Table 2. Securable peripherals by TZSC (continued)

Bus	Peripheral
AHB1	ADF1
	ICACHE registers
	RAMCFG
	TSC
	CRC
APB3	COMP
	LPTIM4
	LPTIM3
	LPTIM1
	I2C3
	LPUART1
APB2	I3C2
	USB
	SAI1
	TIM17
	TIM16
	TIM15
	USART1
	SPI1
	TIM1

Table 2. Securable peripherals by TZSC (continued)

Bus	Peripheral
APB1	VREFBUF
	OPAMP
	I3C1
	FDCAN1
	LPTIM2
	CRS
	I2C2
	I2C1
	UART5
	UART4
	USART3
	SPI3
	SPI2
	IWDG
	WWDG
	TIM7
	TIM6
	TIM4
	TIM3
	TIM2

Table 3. TrustZone aware peripherals

Bus	Peripheral
AHB2	GPIOH
	GPIOG
	GPIOE
	GPIOD
	GPIOC
	GPIOB
	GPIOA

Table 3. TrustZone aware peripherals (continued)

Bus	Peripheral
AHB1	GTZC1
	ICACHE
	FLASH
	EXTI
	GPDMA1
	RCC
	PWR
APB3	SYSCFG
APB1	TAMP
	RTC

2.3 Memory organization

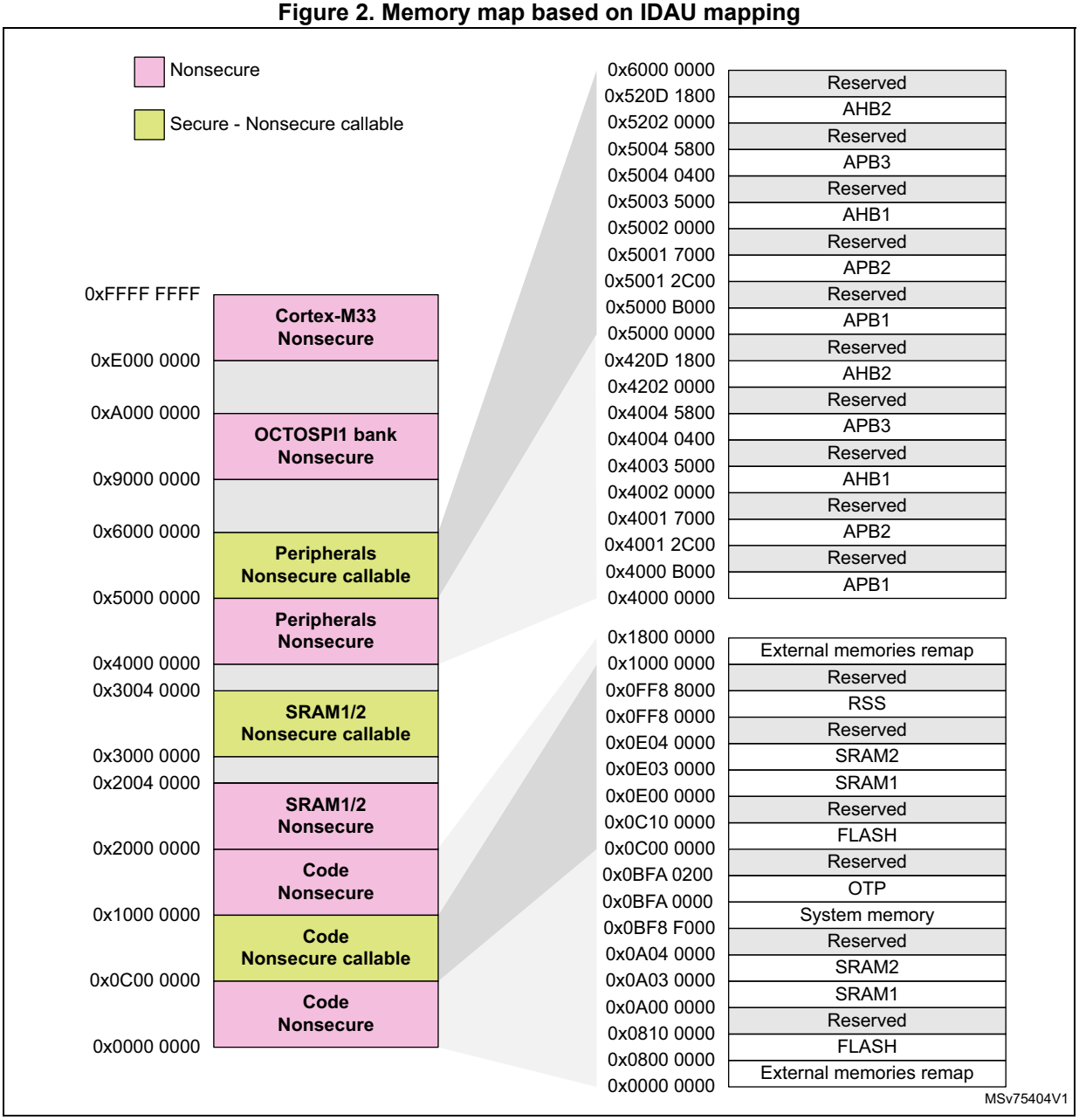
2.3.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map based on IDAU mapping



All memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. The following table gives the boundary addresses of the peripherals available in the devices.

Table 4. Memory map and peripheral register boundary addresses

Bus	Secure boundary address	Nonsecure boundary address	Size (bytes)	Peripheral	Peripheral register map section number
AHB2	0x520D 1800 - 0x5FFF FFFF	0x420D 1800 - 0x4FFF FFFF	223 M	Reserved	-
	0x520D 1400 - 0x520D 17FF	0x420D 1400 - 0x420D 17FF	1 K	OCTOSPI1	*19.7.28*
	0x520C F400 - 0x520D 13FF	0x420C F400 - 0x420D 13FF	8 K	Reserved	-
	0x520C F000 - 0x520C F3FF	0x420C F000 - 0x420C F3FF	1 K	DLYBOS1	*21.5.3*
	0x520C 8800 - 0x520C EFFF	0x420C 8800 - 0x420C EFFF	26 K	Reserved	-
	0x520C 8400 - 0x520C 87FF	0x420C 8400 - 0x420C 87FF	1 K	DLYBSD1	*21.5.3*
	0x520C 8000 - 0x520C 83FF	0x420C 8000 - 0x420C 83FF	1 K	SDMMC1	*20.10.21*
	0x520C 7C00 - 0x520C 7FFF	0x420C 7C00 - 0x420C 7FFF	1 K	CCB	*29.7.4*
	0x520C 4000 - 0x520C 7BFF	0x420C 4000 - 0x420C 7BFF	15 K	Reserved	-
	0x520C 2000 - 0x520C 3FFF	0x420C 2000 - 0x420C 3FFF	8 K	PKA +RAM	*34.8.5*
	0x520C 1000 - 0x520C 1FFF	0x420C 1000 - 0x420C 1FFF	4 K	Reserved	-
	0x520C 0C00 - 0x520C 0FFF	0x420C 0C00 - 0x420C 0FFF	1 K	SAES	*32.8.21*
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	1 K	RNG	*30.7.6*
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	1 K	HASH	*33.6.8*
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	1 K	AES	*31.8.21*
	0x5202 8800 - 0x520B FFFF	0x4202 8800 - 0x420B FFFF	606 K	Reserved	-
	0x5202 8400 - 0x5202 87FF	0x4202 8400 - 0x4202 87FF	1 K	DAC1	*23.7.22*
	0x5202 8300 - 0x520283FF	0x4202 8300 - 0x4202 83FF	256	ADCC12	*22.8.1*
	0x5202 8200 - 0x5202 82FF	0x4202 8200 - 0x4202 82FF	256	Reserved	-
	0x5202 8100 - 0x5202 81FF	0x4202 8100 - 0x4202 81FF	256	ADC2	*22.7.28*
	0x5202 8000 - 0x5202 80FF	0x4202 8000 - 0x4202 80FF	256	ADC1	*22.7.3*
	0x5202 2000 - 0x5202 7FFF	0x4202 2000 - 0x4202 7FFF	24 K	Reserved	-
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	1 K	GPIOH	*12.6.14*
	0x5202 1800 - 0x5202 1BFF	0x4202 1800 - 0x4202 1BFF	1 K	GPIOG	
	0x5202 1400 - 0x5202 17FF	0x4202 1400 - 0x4202 17FF	1 K	Reserved	-
	0x5202 1000 - 0x5202 13FF	0x4202 1000 - 0x4202 13FF	1 K	GPIOE	*12.6.14*
	0x5202 0C00 - 0x5202 0FFF	0x4202 0C00 - 0x4202 0FFF	1 K	GPIOD	
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	1 K	GPIOC	
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	1 K	GPIOB	
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	1 K	GPIOA	

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Size (bytes)	Peripheral	Peripheral register map section number
APB3	0x5004 5800 - 0x5201 FFFF	0x4004 5800 - 0x4201 FFFF	31 M	Reserved	-
	0x5004 5400 - 0x5004 57FF	0x4004 5400 - 0x4004 57FF	1 K	COMP	*25.6.3*
	0x5004 5000 - 0x5004 53FF	0x4004 5000 - 0x4004 53FF	1 K	Reserved	-
	0x5004 4C00 - 0x5004 4FFF	0x4004 4C00 - 0x4004 4FFF	1 K	LPTIM4	*39.7.19*
	0x5004 4800 - 0x5004 4BFF	0x4004 4800 - 0x4004 4BFF	1 K	LPTIM3	
	0x5004 4400 - 0x5004 47FF	0x4004 4400 - 0x4004 47FF	1 K	LPTIM1	
	0x5004 2C00 - 0x5004 43FF	0x4004 2C00 - 0x4004 43FF	6 K	Reserved	-
	0x5004 2800 - 0x5004 2BFF	0x4004 2800 - 0x4004 2BFF	1 K	I2C3	*45.9.13*
	0x5004 2400 - 0x5004 27FF	0x4004 2400 - 0x4004 27FF	1 K	LPUART1	*48.7.15*
	0x5004 0800 - 0x5004 23FF	0x4004 0800 - 0x4004 23FF	7 K	Reserved	-
	0x5004 0400 - 0x5004 07FF	0x4004 0400 - 0x4004 07FF	1 K	SYSCFG	*13.3.11*
	0x5004 0000 - 0x5004 03FF	0x4004 0000 - 0x4004 03FF	1 K	Reserved	-
AHB1	0x5003 5000 - 0x5003 FFFF	0x4003 5000 - 0x4003 FFFF	44 K	Reserved	-
	0x5003 4000 - 0x5003 4FFF	0x4003 4000 - 0x4003 4FFF	4 K	ADF1	*27.8.17*
	0x5003 3400 - 0x5003 3FFF	0x4003 3400 - 0x4003 3FFF	3 K	Reserved	-
	0x5003 3000 - 0x5003 33FF	0x4003 3000 - 0x4003 33FF	1 K	GTZC1 MPCBB2	*5.5.5*
	0x5003 2C00 - 0x5003 2FFF	0x4003 2C00 - 0x4003 2FFF	1 K	GTZC1 MPCBB1	*5.4.5*
	0x5003 2800 - 0x5003 2BFF	0x4003 2800 - 0x4003 2BFF	1 K	GTZC1 TZIC	*5.3.13*
	0x5003 2400 - 0x5003 27FF	0x4003 2400 - 0x4003 27FF	1 K	GTZC1 TZSC	*5.2.8*
	0x5003 2000 - 0x5003 23FF	0x4003 2000 - 0x4003 23FF	1 K	EXTI	*17.4.12*
	0x5003 1000 - 0x5003 1FFF	0x4003 1000 - 0x4003 1FFF	4 K	Reserved	-
	0x5003 0C00 - 0x5003 0FFF	0x4003 0C00 - 0x4003 0FFF	1 K	RCC	*10.5.1*
	0x5003 0800 - 0x5003 0BFF	0x4003 0800 - 0x4003 0BFF	1 K	PWR	*9.5.29*
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	1 K	ICACHE	*8.7.8*
	0x5002 7000 - 0x5003 03FF	0x4002 7000 - 0x4003 03FF	37 K	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	4 K	RAMCFG	*6.6.12*
	0x5002 4400 - 0x5002 5FFF	0x4002 4400 - 0x4002 5FFF	7 K	Reserved	-
	0x5002 4000 - 0x5002 43FF	0x4002 4000 - 0x4002 43FF	1 K	TSC	*28.6.11*
	0x5002 3400 - 0x5002 3FFF	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	1 K	CRC	*18.4.6*
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	-

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Size (bytes)	Peripheral	Peripheral register map section number
AHB1	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	1 K	FLASH	*7.10.1*
	0x5002 1000 - 0x5002 1FFF	0x4002 1000 - 0x4002 1FFF	4 K	Reserved	-
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	4 K	GPDMA1	*15.8.20*
APB2	0x5001 7000 - 0x5001 FFFF	0x4001 7000 - 0x4001 FFFF	36 K	Reserved	-
	0x5001 6C00 - 0x5001 6FFF	0x4001 6C00 - 0x4001 6FFF	1 K	I3C2	*46.16.30*
	0x5001 6400 - 0x5001 6BFF	0x4001 6400 - 0x4001 6BFF	2 K	USB RAM	*52.6.8*
	0x5001 6000 - 0x5001 63FF	0x4001 6000 - 0x4001 63FF	1 K	USB	
	0x5001 5800 - 0x5001 5FFF	0x4001 5800 - 0x4001 5FFF	2 K	Reserved	-
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	1 K	SAI1	*50.6.19*
	0x5001 4C00 - 0x5001 53FF	0x4001 4C00 - 0x4001 53FF	2 K	Reserved	-
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*38.8.22*
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x5001 4000 - 0x5001 43FF	0x4001 4000 - 0x4001 43FF	1 K	TIM15	*38.7.23*
	0x5001 3C00 - 0x5001 3FFF	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*47.8.17*
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	1 K	Reserved	-
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	1 K	SPI1	*49.8.15*
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	*35.6.31*
	0x5001 0000 - 0x5001 2BFF	0x4001 0000 - 0x4004 2BFF	11 K	Reserved	-
APB1	0x5000 B000 - 0x5000 FBFF	0x4000 B000 - 0x4000 FFFF	20 K	Reserved	-
	0x5000 AC00 - 0x5000 AFFF	0x4000 AC00 - 0x4000 AFFF	1 K	FDCAN1 SRAM	-
	0x5000 A800 - 0x5000 ABFF	0x4000 A800 - 0x4000 ABFF	1 K	Reserved	-
	0x5000 A400 - 0x5000 A7FF	0x4000 A400 - 0x4000 A7FF	1 K	FDCAN1	*51.4.38*
	0x5000 9800 - 0x5000 A3FF	0x4000 9800 - 0x4000 A3FF	3 K	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2	*39.7.19*
	0x5000 8000 - 0x5000 93FF	0x4000 8000 - 0x4000 93FF	5 K	Reserved	-
	0x5000 7C00 - 0x5000 7FFF	0x4000 7C00 - 0x4000 7FFF	1 K	TAMP	*44.6.15*
	0x5000 7800 - 0x5000 7BFF	0x4000 7800 - 0x4000 7BFF	1 K	RTC	*43.6.28*
	0x5000 7400 - 0x5000 77FF	0x4000 7400 - 0x4000 77FF	1 K	VREFBUF	*24.5.3*
	0x5000 7000 - 0x5000 73FF	0x4000 7000 - 0x4000 73FF	1 K	OPAMP	*26.5.7*
	0x5000 6400 - 0x5000 6FFF	0x4000 6400 - 0x4000 6FFF	3 K	Reserved	-
	0x5000 6000 - 0x5000 63FF	0x4000 6000 - 0x4000 63FF	1 K	CRS	*11.7.5*

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Size (bytes)	Peripheral	Peripheral register map section number
APB1	0x5000 5C00 - 0x5000 5FFF	0x4000 5C00 - 0x4000 5FFF	1 K	I3C1	*46.16.30*
	0x5000 5800 - 0x5000 5BFF	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	*45.9.13*
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	1 K	I2C1	
	0x5000 5000 - 0x5000 53FF	0x4000 5000 - 0x4000 53FF	1 K	UART5	*47.8.17*
	0x5000 4C00 - 0x5000 4FFF	0x4000 4C00 - 0x4000 4FFF	1 K	UART4	
	0x5000 4800 - 0x5000 4BFF	0x4000 4800 - 0x4000 4BFF	1 K	USART3	
	0x5000 3C00 - 0x5000 47FF	0x4000 3C00 - 0x4000 47FF	3 K	Reserved	-
	0x5000 3800 - 0x5000 3BFF	0x4000 3800 - 0x4000 3BFF	1 K	SPI2	*49.8.15*
	0x5000 3400 - 0x5000 37FF	0x4000 3400 - 0x4000 37FF	1 K	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*41.7.7*
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	*42.6.4*
	0x5000 2400 - 0x5000 2BFF	0x4000 2400 - 0x4000 2BFF	2 K	Reserved	-
	0x5000 2000 - 0x5000 23FF	0x4000 2000 - 0x4000 23FF	1 K	SPI3	*49.8.15*
	0x5000 1800 - 0x5000 1FFF	0x4000 1800 - 0x4000 1FFF	2 K	Reserved	-
	0x5000 1400 - 0x5000 17FF	0x4000 1400 - 0x4000 17FF	1 K	TIM7	*37.4.9*
	0x5000 1000 - 0x5000 13FF	0x4000 1000 - 0x4000 13FF	1 K	TIM6	
	0x5000 0C00 - 0x5000 0FFF	0x4000 0C00 - 0x4000 0FFF	1 K	Reserved	-
	0x5000 0800 - 0x5000 0BFF	0x4000 0800 - 0x4000 0BFF	1 K	TIM4	*36.5.25*
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	1 K	TIM3	
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	1 K	TIM2	

2.3.3 Embedded SRAMs

The devices feature up to 256-Kbyte SRAMs:

- 192-Kbyte SRAM1
- 64-Kbyte SRAM2

These SRAMs can be accessed as bytes, half-words (16 bits), or full words (32 bits). These memories can be addressed both by CPU and DMAs.

The CPU can access the SRAM1 and SRAM2 through the system bus or through the C-bus depending on the selected address.

When the TrustZone security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as nonsecure with a block granularity. For more details, refer to *Section 5: Global TrustZone controller (GTZC)*.

SRAM features are detailed in *Section 6.3.1: Internal SRAMs features*.

2.3.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- the main flash memory block, that contains the application program and user data
- the information block, that is composed of the following parts:
  - option bytes for hardware and memory protection user configuration
  - system memory that contains ST proprietary code
  - OTP (one-time programmable) area

The flash memory interface implements instruction access and data access based on the AHB protocol. It also implements the logic necessary to carry out the flash memory operations (program/erase) controlled through the flash memory registers plus security access control features. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

3 System security

The STM32U3 series devices are designed with a comprehensive set of security features, some of which being based on the standard Arm TrustZone technology.

These security features simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEM and third-party developers, by facilitating the reuse, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on STM32U3 series devices.

3.1 Key security features

- Resource isolation using privilege mode and Armv8-M mainline security extension of the Cortex-M33, extended to securable I/Os, memories, and peripherals
- Secure firmware installation (SFI) with device unique cryptographic key pair
  - leveraging the on-chip immutable bootloader that supports the download of the image through USART, USB, I2C, I3C, SPI, FDCAN, and JTAG
- Secure boot thanks to the unique boot entry feature and hide-protect area (HDP) mechanism
- Secure storage, featuring:
  - Nonvolatile on-chip secure storage, protected with secure and HDP areas
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID and JTAG 32-bit device-specific ID
  - On-chip enhance storage technology, using hardware secret nonvolatile derived hardware unique keys (DHUK), and application-defined volatile boot hardware key (BHK), both loadable by hardware to the DPA-resistant SAES engine
- General purpose cryptographic acceleration:
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
  - Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM and CCM chaining modes with side-channel counter-measures and mitigations
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B pre-certified
  - Coupling and chaining bridge (CCB) for special coupling and chaining operations (involving PKA, SAES, RNG peripherals) to protect private keys used in PKA protected operations.
- Flexible life-cycle scheme with readout protection (RDP), including support for product decommissioning (auto-erase)
  - Debug protection, depending on the RDP level
  - Optional password-based RDP level regressions, including for RDP level 2
- Protected firmware distribution scheme, using TrustZone and RDP level 0.5

- External tamper detection and protection against frequency attacks
  - Five passive inputs, available in all power modes

3.2 Secure install

The secure firmware install (SFI) is an STMicroelectronics secure service authenticated and decrypted by the immutable RSS code stored in the device. The SFI allows secure and counted installation of OEM firmware in the untrusted production environment (such as the OEM contract manufacturer).

The confidentiality of the installed images written in the internal flash memory, is also protected, using the AES.

The SFI native service leverages the following hardware security features:

- secure boot (see *Section 3.3*)
- resource isolation using TrustZone (see *Section 3.5*)
- temporal isolation using hide protection (see *Section 3.6.1*)
- secure execution (see *Section 3.7*)
- secure storage, with associated cryptographic engines (see *Section 3.8* and *Section 3.9*)

Further information can be found in the application note: *Overview secure firmware install (SFI) (*AN4992).

3.3 Secure boot

Secure boot is an immutable code that is always executed after a system reset. As a root of trust, this code checks the device static protections and activates available device runtime protections, reducing the risk that invalid or malicious code runs on the platform. As root of trust, the secure boot also checks the integrity and authenticity of the next level firmware before executing it.

The actual functions of the secure boot depend on the availability of TrustZone features, and on the firmware stored in the device. However, the secure boot typically initializes the secure storage.

The device trusted firmware-M (TFM) application, supported by the STM32 ecosystem, provides a root of trust solution including secure boot functions. For more information, refer to the user manual *Getting started with STM32CubeU5 TFM application* (UM2851).

In the devices, the secure boot takes benefit of hardware security features such as:

- resource isolation using TrustZone (see *Section 3.5*)
- temporal isolation using hide protection (see *Section 3.6.1*)
- secure execution (see *Section 3.7*)
- secure install and update (see *Section 3.2* and *Section 3.4*)
- secure storage, with associated cryptographic engines if available (see *Section 3.8* and *Section 3.9*)

This section describes the features specifically designed for secure boot.

3.3.1 Unique boot entry and BOOT_LOCK

When TrustZone is activated (TZEN = 1) and the BOOT_LOCK secure option bit is cleared, the application selects a boot entry point located either in the system flash memory (see the next section), or in the secure user flash memory, at the address defined by option bytes of FLASH_SBOOT0R.

When TrustZone is activated (TZEN = 1) and the BOOT_LOCK secure option bit is set, the device unique boot entry is the unmodifiable secure address defined by of FLASH_SBOOT0R option bytes. All these option bytes cannot be modified by the application anymore when BOOT_LOCK is set.

Note: As long as it is cleared, the BOOT_LOCK option bit can be set without any constraint. But once set, the BOOT_LOCK option bit cannot be cleared when RDP level > 0.

For more information on the boot mechanisms, refer to *Section 4: Boot modes*.

3.3.2 Immutable root of trust in system flash memory

The immutable root-of-trust code stored in the system flash memory is first used to initiate SFI, allowing secure and counted installation of OEM firmware in untrusted production environment (such as OEM contract manufacturer).

The STMicroelectronics immutable code also includes secure runtime services that can be called at runtime when a secure application sets the SYSCGFG_RSSCMDR register to a non-null value before triggering a system reset. This runtime feature is deactivated when the BOOT_LOCK secure option bit is set, and the secure address defined by option bytes of FLASH_SBOOT0R are set on the secure user flash memory.

3.4 Secure update

The secure firmware update is a secure service that runs after a secure boot. Its actual functions depend on the availability of the TrustZone features, and on the firmware stored in the device.

The device trusted firmware-M (TFM) application, supported by the STM32 ecosystem, allows the update of the microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way to prevent unauthorized updates and access to confidential on-device data.

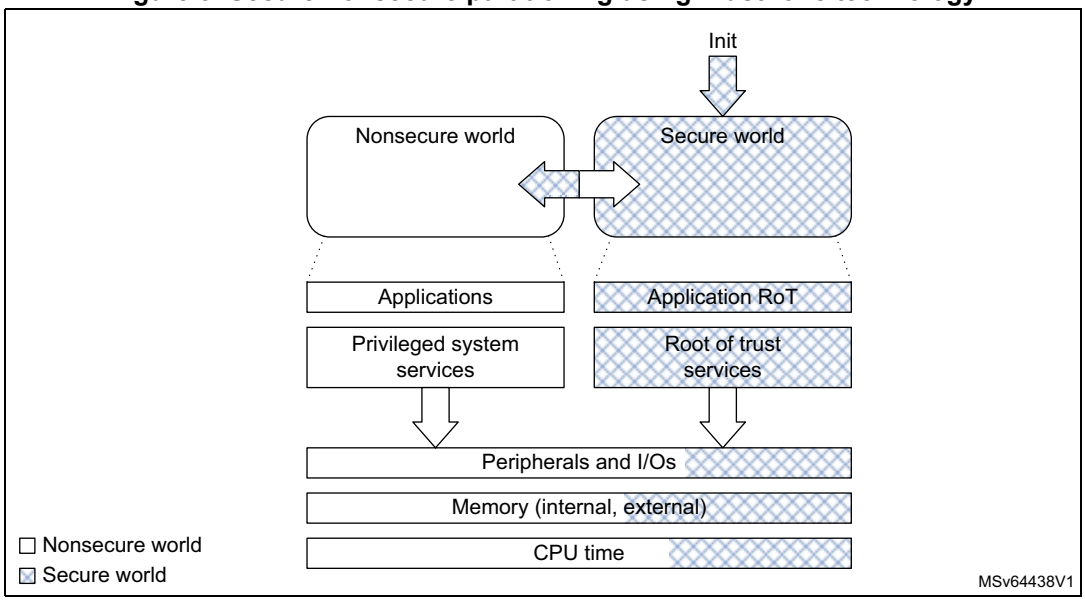
A firmware update can be done either on a single firmware image including both secure and nonsecure parts, or on the secure (respectively nonsecure) part of the firmware image, independently.

In the devices, the secure update application leverages the same hardware security as the firmware install described in *Section 3.2*. For more information, refer to the user manual: *Getting started with STM32CubeU5 TFM application* (UM2851).

3.5 Resource isolation using TrustZone

In the STM32U3 series devices, the hardware and software resources can be partitioned so that they exist either in the secure world or in the nonsecure world, as shown in the figure below.

Figure 3. Secure/nonsecure partitioning using TrustZone technology



Note: The initial partitioning of the platform is under the responsibility of the secure firmware executed after the reset of the device.

Thanks to this resource isolation technology, the secure world can be used to protect critical code against intentional or unintentional tampering from the more exposed code running in the nonsecure world.

Note: The secure code is typically small and rarely modified, while nonsecure code is more exposed, and prone to firmware updates.

3.5.1 TrustZone security architecture

The Armv8-M TrustZone technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. The device TrustZone hardware features include:

- the Armv8-M mainline security extension of Cortex-M33, enabling a new processor secure state, with its associated secure interrupts
- the dynamic allocation of memory and peripherals to TrustZone using eight security attribution unit (SAU) regions of Cortex-M33
- a global TrustZone framework (GTZC), extending the TrustZone protection against transactions coming from other masters in the system than the Cortex-M33
- TrustZone-aware embedded flash memory and peripherals

Note: The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

3.5.2 Armv8-M security extension of Cortex-M33

The Arm security extension of the Cortex-M33 is an evolution, not a revolution. It uses the programmer model from earlier Cortex-M subfamilies like Cortex-M4. Indeed, Armv8-M is architecturally similar to Armv7-M, using the same 32-bit architecture, the same memory mapped resources protected with an MPU. Armv8-M also uses the nested vectored interrupt controller (NVIC).

The Armv8-M TrustZone implementation in STM32U3 series devices is composed of the following features:

- a new processor state, with almost no additional code/cycle overhead, as opposed to Armv8-A TrustZone that uses a dedicated exception routine for triggering a secure/nonsecure world change
- two memory map views of a shared 4-Gbyte address space
- a low interrupt latency for both secure and nonsecure domains, and a new interrupt configuration for security grouping and priority setting
- separated exception vector tables for the secure and nonsecure exceptions
- micro-coded context preservation
- banking of specific registers across secure/nonsecure states, including stack pointers with stack-limit checkers
- banking of the following Cortex-M33 programmable components (two separate units for secure and nonsecure):
  - SysTick timer
  - MPU configuration registers (twelve MPU regions in secure, eight in nonsecure)
  - some of the system control block (SCB) registers
- new system exception (SecureFault) for handling of security violations
- configurable debug support, as defined in *Section 3.11*

For more information, refer to *STM32 Cortex-M33 MCUs programming manual* (PM0264).

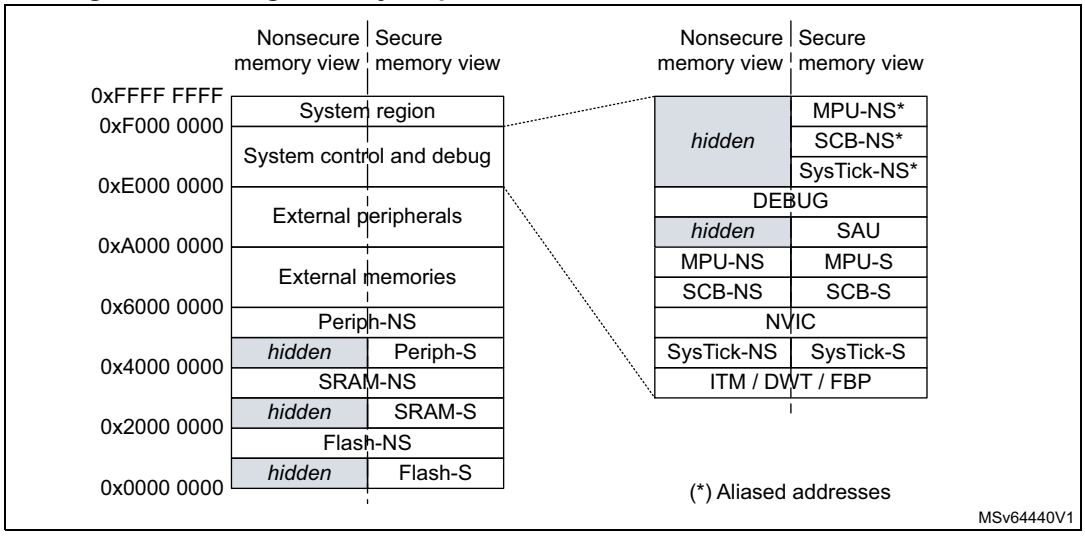
3.5.3 Memory and peripheral allocation using IDAU/SAU

Security attributes

As illustrated on *Figure 4*, the Armv8-M nonsecure memory view is similar to Armv7-M (that can be found in Cortex-M4), with the difference that the secure memory is hidden. The secure memory view shows the flash memory, SRAM, and peripherals that are only accessible while the Cortex processor executes in Secure state.

Figure 4 shows the 32-bit address space viewed after the SAU configuration by the secure code.

Figure 4. Sharing memory map between CPU in secure and nonsecure state

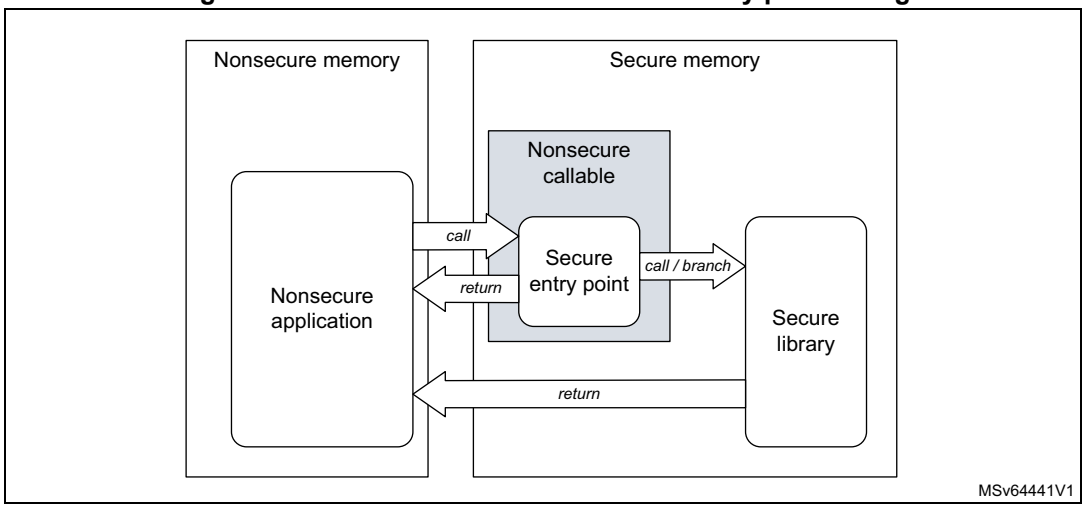


The Cortex processor state (and associated rights) depends on the security attribute assigned to the memory region where it is executed:

- A processor in a nonsecure state only executes from nonsecure (NS) program memory, while a processor in a secure state only executes from secure (S) program memory.
- While running in the secure state, the processor can access data from both S and NS memories. Running in the nonsecure state, the CPU is limited to nonsecure memories.

In order to manage transitions to the secure world, developers must create nonsecure callable (NSC) regions that contain valid entry points to the secure libraries. The first instruction in these entry points must be the new secure gate (SG) instruction, used by the nonsecure code to call a secure function (see *Figure 5*).

Figure 5. Secure world transition and memory partitioning



Programming security attributes

In Cortex-M33, the static implementation defined attribution unit (IDAU) works in conjunction with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in the table below.

Table 5. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Nonsecure	Secure	Secure
	Secure-NSC	Secure-NSC
	Nonsecure	Nonsecure
Secure-NSC	Secure	Secure
	Nonsecure	Secure-NSC

1. Defined regions are aligned to 32-byte boundaries.

The SAU can only be configured by the Cortex-M33 in the secure-privilege state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in *Table 5*.

Note: The SAU/IDAU settings are applicable only to the Cortex-M33*. The other masters like DMA are not affected by these policies.*

A memory space not covered by an SAU region is fixed as secure.

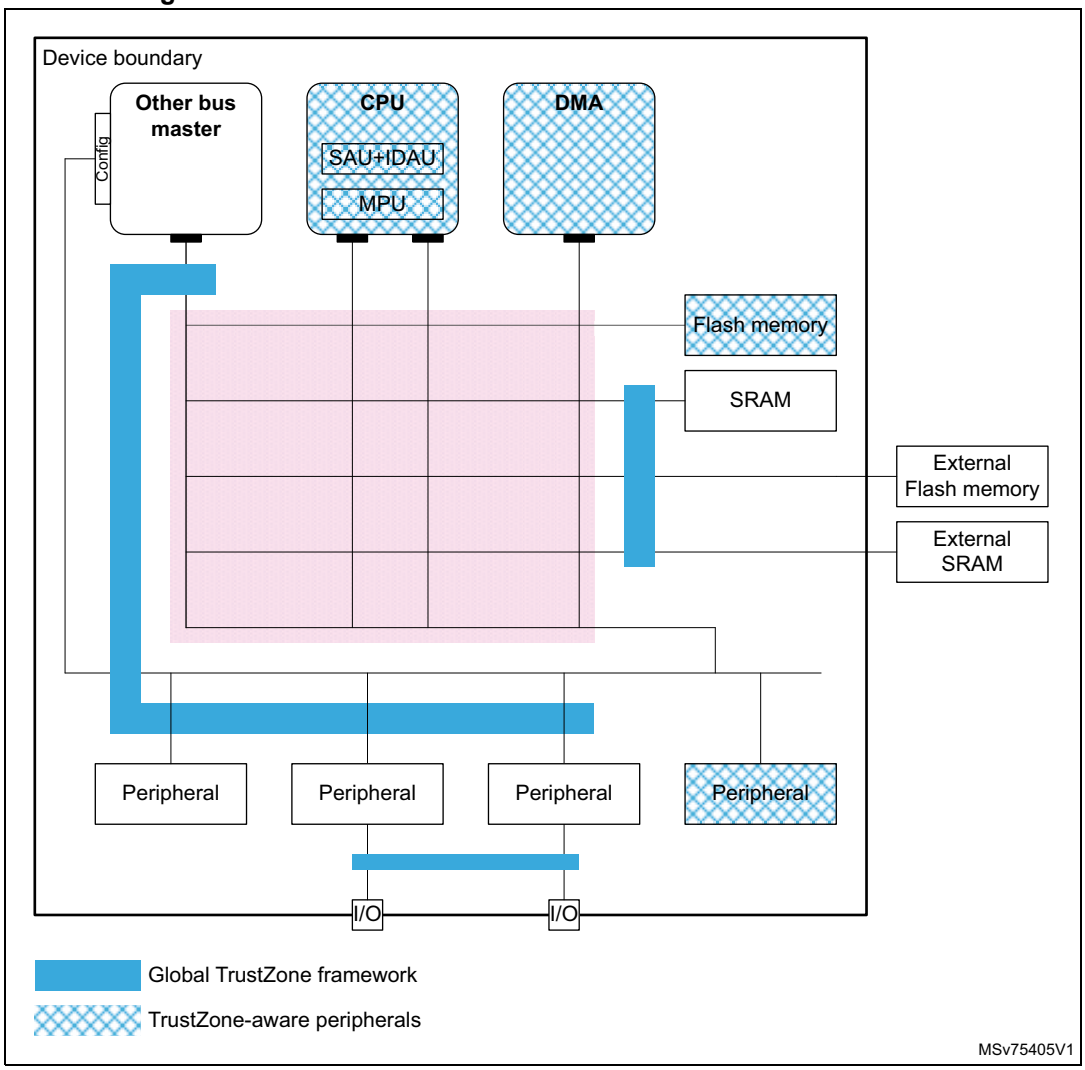
For more information on memory security attribution using IDAU/SAU, refer to the application note *Arm TrustZone features for STM32L5, STM32U3, and STM32U5 series* (AN5347).

3.5.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed complementary security features that reinforce, in a flexible way, the isolation between the secure and the nonsecure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against nonsecure transactions coming from other masters than the Cortex-M33.

Figure 6. Global TrustZone framework and TrustZone awareness



Securing peripherals

When the TrustZone security is active, a peripheral is either securable through the TZSC in GTZC, or is natively TrustZone-aware, as shown in *Figure 6*:

- A securable peripheral or memory is protected by an AHB/APB firewall gate, that is controlled by the TrustZone security controller (TZSC).
- A TrustZone-aware peripheral or memory is connected directly to the AHB or APB interconnect, implementing a specific TrustZone behavior, such as a subset of secure registers, or a secure memory area.

When a securable peripheral is made secure-only with the GTZC, if this peripheral is master on the interconnect (such as SDMMC), it automatically issues secure transactions. The SDMMC is an example of securable master. TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently to the GTZC.

Note: Like with TrustZone, a peripheral can be made privileged-only with TZSC (see Section 3.6.2). In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions.

Securing memories with TZSC and MPCBB

The MPCBB resources in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in the following table.

Table 6. MPCBBx resources

Memory	MPC resource	Type of filtering	Memory size (Kbytes)	Block size (Bytes)	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block based, managing security and privilege	192	512(1)	12	Secure privileged(2)
SRAM2	GTZC1_MPCBB2		64		4	

1. Blocks are grouped in super-blocks of 32 consecutive blocks, to manage the configuration locking.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

Applying GTZC configurations

The TZSC and MPCBB blocks can be used in one of the following ways:

- statically programmed during the secure boot, locked and not changed afterwards
- dynamically reprogrammed using a specific application code or real-time kernel

When the dynamic option is selected and the configuration is not locked:

- MPCBB secure blocks size can be changed by a secure software. This software can be unprivileged if the particular block is not privileged-only.
- The secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC_TZSC_PRIVCFGRx) registers.

Securing peripherals with TZSC

The TZSC block in GTZC provides the capability to manage the security and the privilege for all securable peripherals. The list of these peripherals can be found in *Section 5: Global TrustZone controller (GTZC)*.

When the TrustZone is activated, peripherals are set as nonsecure and unprivileged after reset.

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only (see Section 3.6.2).

-

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOH
- GTZCx_MPCBB, GTZCx_TZIC and GTZCx_TZSC (GTZC blocks)
- EXTI
- Flash memory
- RCC and PWR
- GPDMA
- SYSCFG registers
- RTC and TAMP
- MCU debug unit DBGMCU

The way illegal accesses to these peripherals are monitored through the TZIC registers is described in *Section 5: Global TrustZone controller (GTZC)*.

For more details, refer to *Section 3.5.5*.

TrustZone illegal access controller (TZIC)

The TZIC block in GTZC gathers all illegal access events originated from sources either protected by GTZC or TrustZone-aware peripherals, generating one global secure interrupt towards the NVIC.

TZIC is available only when the system is TrustZone enabled (TZEN = 1). All accesses to TZIC registers must be secured and privileged.

For each illegal event source, a status flag and a clear bit exist. Each illegal event can be masked, not generating an interrupt toward the NVIC.

Note: By default, all events are masked.

3.5.5 Managing security in TrustZone-aware peripherals

This section gives more details on how the security is implemented in the TrustZone-aware peripherals listed in the previous section.

Embedded flash memory

When the TrustZone security is enabled through option bytes (TZEN = 1), the whole flash memory is secure after reset and the following protections, shown in *Figure 7*, are available to the application:

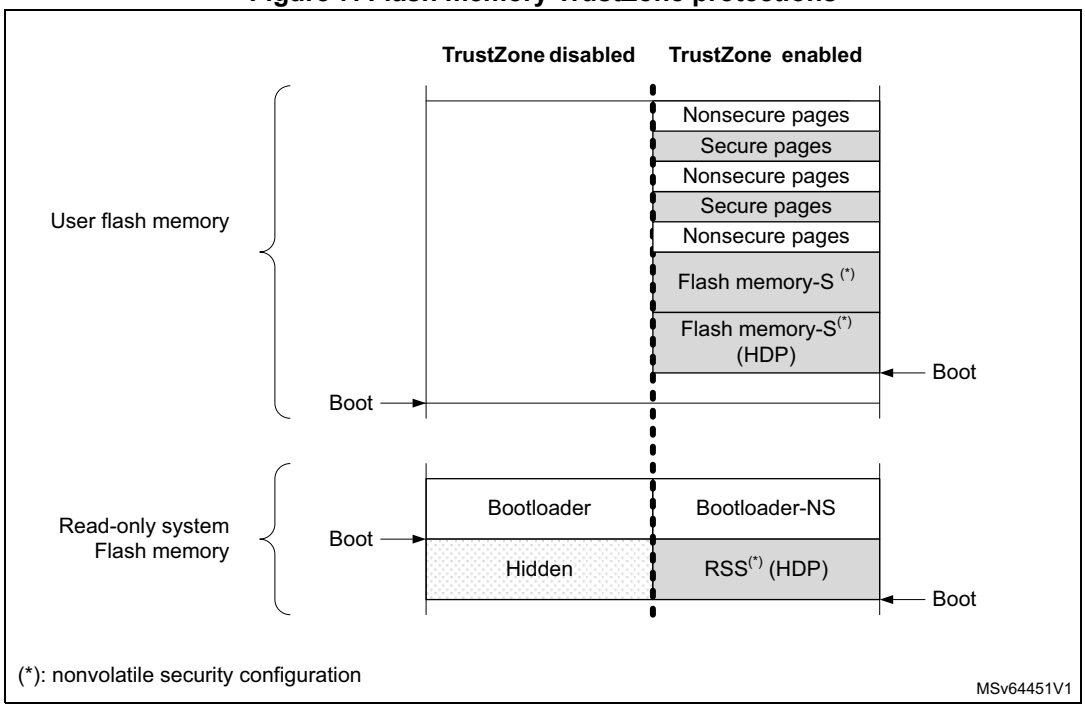
- nonvolatile user secure areas, defined with nonvolatile secure user option bytes
  - watermark-based secure only area (x2)
  - secure hide protection (HDP) area, stickily hidden after boot (x2)
- volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set as nonsecure (example: outside the watermark-based secure only area), can be set as secure on-the-fly using the block-based configuration registers.

Note: All areas are aligned on the flash memory page granularity.

The flash memory area can be configured as secure while it is tagged as nonsecure in Cortex-M33 IDAU/SAU. In this case, nonsecure accesses by the CPU to the flash memory are denied.

Erase or program operations can be available to secure (resp. nonsecure) code only for secure (resp. nonsecure) pages or memory. A flash memory is considered secure if at least one page is secure.

Figure 7. Flash memory TrustZone protections



As shown in *Section 7: Embedded flash memory (FLASH)*, when TrustZone is activated (TZEN = 1), the application code can use the HDP area that is part of the flash memory watermark-based secure area. Indeed, when the application modifies the HDPx_ACCDIS byte to a value different than the reset value, data read, write, and instruction fetch on this HDP area are denied until the next system reset.

For example, the software code in the secure flash memory HDP area can be executed only once, with any further access to this area denied until the next system reset. Additionally, any flash memory page belonging to an active HDP area cannot be erased anymore.

When the TrustZone is deactivated (TZEN = 0), the volatile/non-volatile secure area features are deactivated and all secure registers are RAZ/WI.

See *Section 7: Embedded flash memory (FLASH)* for more details.

Direct memory access controllers (GPDMA)

When a DMA channel x is defined as secure (SECx = 1 in GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or nonsecure by a secure application using SSEC and DSEC bits in GPDMA_CxTR1.

The table below summarizes these security options available in each DMA channel.

Table 7. DMA channel use (security)(1)

Destination type	Secure DMA channel x (SECx = 1)		Nonsecure DMA channel y (SECy = 0)	
	Secure source	Nonsecure source	Secure source	Nonsecure source
Secure destination	OK	OK(2)	Transfer blocked	
Nonsecure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return zeros. Also an illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.
2. If the source is a memory, the transfer is only possible if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory, the transfer is only possible if DSEC = 0, otherwise the transfer is blocked.
4. If the transfer is memory-to-memory, the transfer is only possible if SSEC = 0 and DSEC = 0, otherwise the transfer is blocked.

When a channel is configured as secure:

- Registers allocated to this channel (excluding GPDMA_SECCFGR, GPDMA_PRIVCFGR, and GPDMA_RCFGLOCKR) are read as zero. Write are ignored for nonsecure accesses. A secure illegal access event may also be triggered toward the TZIC peripheral.
  - Writes to GPDMA_SECCFGR and GPDMA_RCFGLOCKR must be secure. For each bit in GPDMA_PRIVCFGR, write must be secure if the corresponding bit in GPDMA_SECCFGR is set.
- In linked-list mode, the loading of the next linked-list data structure from memory is performed with secure transfers.
- When switching to a nonsecure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

Note: DMA secure channels are not available when TrustZone is deactivated.

When a channel is configured as nonsecure, in linked-list mode, the loading of the next linked-list data structure from memory is performed with nonsecure transfers.

See *Section 15: General purpose direct memory access controller (GPDMA)* for more details.

Power control (PWR)

When the TrustZone security is activated (TZEN = 1), the selected PWR registers can be secured through PWR_SECCFGR, protecting the following PWR features:

- low-power mode setup
- wake-up (WKUP) pins definition
- voltage detection and monitoring
- backup domain control

Other PWR configuration bits become secure:

- when the system clock selection is secure in the RCC: the voltage scaling (VOS) and the regulator booster (BOOSTEN) configurations become secure.
- when a GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode becomes secure.

See *Section 9: Power control (PWR)* for details.

Secure clock and reset (RCC)

When the TrustZone security is activated (TZEN = 1) and security is enabled in the RCC, the bits controlling the peripheral clocks and resets become TrustZone-aware:

- If the peripheral is securable and programmed as secure in the TZSC, the peripheral clock and reset bits become secure.
- If the peripheral is TrustZone-aware, the peripheral clock and reset bits become secure as soon as at least one function is configured as secure inside the peripheral.

Note: Refer to Section 3.5.4 for the list of securable and TrustZone-aware peripherals.

Additionally, the following configurations can be made secure-only using RCC_SECCFGR:

- External clock (such as HSE or LSE), internal oscillator (such as HSI, MSI, or LSI)
- AHB prescaler
- System clock source selection
- Reset flag clearing
- Automatic internal oscillator waking up configuration

See *Section 10: Reset and clock control (RCC)* for details.

Real time clock (RTC)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured RTC register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the RTC illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes. The secure boot code can then change this security setup, making registers alarm A, alarm B, wake-up timer, and timestamp secure or not, using RTC_SECCFGR.

When the SEC bit is set in secure-only RTC_SECCFGR:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER, and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC is cleared in secure-only RTC_SECCFGR, it is still possible to restrict access in secure mode to some RTC registers by setting dedicated control bits: INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC, and ALRBSEC.

Note: The RTC security configuration is not affected by a system reset.

See *Section 43: Real-time clock (RTC)* for more details.

Tamper and backup registers (TAMP)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured TAMP register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the TAMP illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes. The secure boot code can change this security setup, making some registers secure or not as needed, using the TAMP_SECCFGR register.

When TAMPSEC is set in TAMP_SECCFGR:

- Writing the TAMP registers is possible only in secure mode. Backup registers have their own write protection (see below).
- Reading the TAMP registers (except for TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR) returns zero if the access is nonsecure. Backup registers have their own read protection (see below).

The application can also:

- make TAMP_COUNTR register read and write secure-only by setting the CNT1SEC bit in TAMP_SECCFGR secure register
- in backup registers increase security for two of the three protection zones configured using BKPRWSEC[7:0] and BKPWSEC[7:0] bitfields in TAMP_SECCFGR:
  - protection zone 1 is read nonsecure, write nonsecure
  - protection zone 2 is read nonsecure, write secure
  - protection zone 3 is read secure, write secure

Note: The TAMP security configuration is not affected by a system reset.

See *Section 44: Tamper and backup registers (TAMP)* for more details.

General-purpose I/Os (GPIO)

When the TrustZone security is activated (TZEN = 1), each I/O pin of the GPIO port can be individually configured as secure through the GPIOx_SECCFGR registers. Only a secure application can write to GPIOx_SECCFGR registers. After boot, each I/O pin of GPIO is set as secure.

When an I/O pin is configured as secure, its corresponding configuration bits for alternate function (AF), mode selection (MODE), and I/O data are RAZ/WI in case of nonsecure access.

When the digital alternate function is used (input/output mode), in order to protect the data transiting from/to the I/O managed by a secure peripheral, the devices add a secure alternate function gate on the path between the peripheral and its allocated I/Os:

- If the peripheral is secure, the I/O pin must also be secure to allow input/output of data.
- If the peripheral is not secure, the connection is allowed regardless of the I/O pin state.

The TrustZone-aware logic around GPIO ports, used as the alternate function, is summarized in the table below.

Table 8. Secure alternate function between peripherals and allocated I/Os

Security configuration		Alternate function logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Nonsecure				Out of reset configuration
Secure	Nonsecure	Zero	Zero	-
Nonsecure		I/O data	Peripheral data	

The following table lists peripherals which, when nonsecure, cannot be connected to secure I/Os configured for an analog function with analog switch, thanks to hardware that blocks such connections.

Table 9. Nonsecure peripherals not connectable to secure I/Os

Peripheral	Analog function(1)	Input	Output	How to set a peripheral or function as secure
ADC1	ADC1_INy	X	-	Set ADC12SEC in GTZC1_TZSC_SECCFGR3
ADC2	ADC2_INy			
OPAMPx	OPAMPx_VINM, OPAMPx_VINP			Set OPAMPSEC in GTZC1_TZSC_SECCFGR1
COMPx	COMPx_INPy, COMPx_INMy			Set COMPSEC in GTZC1_TZSC_SECCFGR2

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).

Finally, regarding GPIO and security, *Table 10* below summarizes the list of peripheral functions that do not have any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral function is nonsecure.

For example, when a secure application sets PA4 as secure to be used as LPTIM2_OUT, if the DAC is nonsecure, it can be programmed to output data to PA4, potentially causing malfunction to the secure application.

Similarly, when a secure application sets PA0 as secure to be used as UART4_TX, if TAMP is nonsecure, it can be programmed to capture the USART input traffic through the TAMP_IN function.

It is important that, for each case described in *Table 10*, the secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential), then the secure application must configure the TAMP as secure even if it is not used by the secure application.

Note: How to make a peripheral secure is summarized in the right column of Table 10.

Table 10. Nonsecure peripherals connectable to secure I/Os

Peripheral	Signal(1)	Input	Output	How to set the peripheral or function as secure
DAC	DAC1_OUTx	-	X	Set DAC1SEC in GTZC1_TZSC_SECCFGR3.
OPAMP	OPAMPx_VOUT	-	X	Set OPAMPSEC in GTZC1_TZSC_SECCFGR1
USB	USB_VBUS	X	X	Set USBSEC in GTZC1_TZSC_SECCFGR2
TAMP	TAMP_INx	X	-	Set TAMPSEC in TAMP_SECCFGR

Table 10. Nonsecure peripherals connectable to secure I/Os (continued)

Peripheral	Signal[1]	Input	Output	How to set the peripheral or function as secure
RTC	RTC_OUTx	-	X	Set SEC in RTC_SECCFGR.
	RTC_TS	X	-	Set TSSEC in RTC_SECCFGR.
PWR	WKUPx	X	-	Set WUPxSEC in PWR_SECCFGR.
RCC	LSCO	-	X	Set LSESEC in RCC_SECCFGR.
EXTI	EXTIx	X	-	Set SECx bit in EXTI_SECCFGR.

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

Refer to *Section 12: General-purpose I/Os (GPIO)* for more details.

Extended interrupts and event controller (EXTI)

When the TrustZone security is activated (TZEN = 1), the EXTI is able to protect event register bits from being modified by nonsecure accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGR1. When an input event is configured as secure, only a secure application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

The security configuration in EXTI_SECCFGR1 can be globally locked after reset in EXTI_LOCKR.

See *Section 17: Extended interrupts and event controller (EXTI)* for more details.

System configuration controller (SYSCFG)

Like all TrustZone-aware peripherals, when the TrustZone security is activated (TZEN = 1), a nonsecure read/write access to a secured SYSCFG register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SYSCFG illegal access event is not masked in the TZIC.

See *Section 13: System configuration controller (SYSCFG)* for more details.

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) helps the debugger, providing support for:

- low-power mode behavior during debug
- peripheral freeze during debug, applicable to I2Cs, IWDG, WWDG, timers, low-power timers, and GPDMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in *Table 11*.

Table 11. TrustZone-aware DBGMCU access management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Nonsecure invasive (SPIDEN = 0)	NS	Yes (S(2) or NS)	Yes (S or NS)
	S	None (S or NS)	
Secure invasive (SPIDEN = 1)	NS	Yes (S or NS)	
	S	Yes (S only)	

1. As reported by the GTZC, the TrustZone-aware peripheral or the DMA channel.
2. Secure access from the debugger is converted to nonsecure access in the device.

Refer to *Section 53: Debug support (DBG)* for more details.

3.5.6 Activating TrustZone security

The TrustZone is deactivated by default in all STM32U3 series devices. It can be activated by setting the TZEN user option bit in FLASH_OPTR when in RDP level 0. Once TZEN has changed from 0 to 1, the default security state, after reset, is always the following:

- CPU subsystem
  - Cortex-M33 exits reset in secure state, hence the boot address must point toward a secure memory area.
  - All interrupt sources are secure (in NVIC).
  - The memory mapped viewed by the CPU through IDAU/SAU is fully secure.
- Embedded flash memory
  - Flash memory nonvolatile secure areas (with their HDP zone), are defined with nonvolatile registers FLASH_SECWMxR (x = 1, 2). Default secure option bytes setup is all user flash secure, without HDP area defined.
  - Volatile block-based security attributions of the flash memory are nonsecure. The secure boot code can change this setup, making the blocks secure.
- Embedded SRAM memories
  - All SRAMs are secure, as defined in GTZC/MPCBB (see *Section 3.5.4*). The secure boot code can change this security setup, making the blocks secure or not.
- External memories are nonsecure
- All GPIOs are secure.
- All GPDMA channels are nonsecure.
- Backup registers are nonsecure.
- Peripherals and GTZC:
  - Securable peripherals are nonsecure and unprivileged.
  - TrustZone-aware peripherals are nonsecure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC/TZIC are disabled.

Note: Refer to Section 3.5.4 for the list of securable and TrustZone-aware peripherals.

3.5.7 Deactivating TrustZone security

Once TrustZone is activated, it can only be deactivated during an RDP regression to level 0.

Note: Such RDP regression triggers the erase of embedded memories (SRAM1, SRAM2, flash memory), and the reset of all peripherals, including all cryptographic engines.

After the TrustZone deactivation, most of the features mentioned in *Section 3.5* are no longer available:

- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- The NVIC only manages nonsecure interrupts.
- All secure registers in TrustZone-aware peripherals are RAZ/WI.

Note: When the TrustZone is deactivated, the resource isolation using privilege stays available (see Section 3.6.2 for details).

For more information, refer to the application note *Arm TrustZone features for STM32L5, STM32U3, and STM32U5 Series* (AN5347).

3.6 Other resource isolations

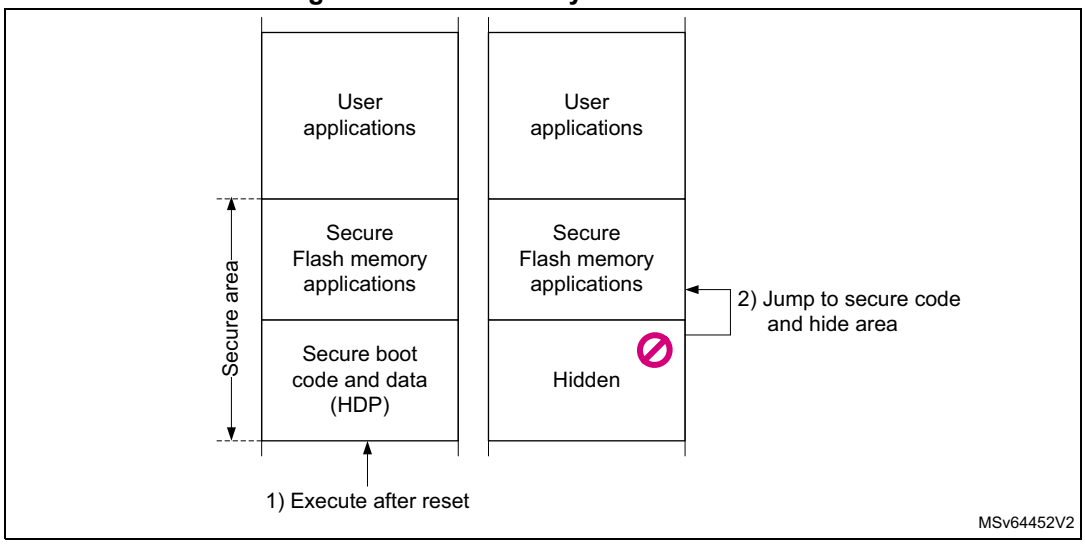
These are hardware mechanisms offering an additional level of isolation on top of the TrustZone technology.

3.6.1 Temporal isolation using secure hide protection (HDP)

When the TrustZone security is enabled (TZEN = 1), the embedded flash memory allows an HDP area per watermarked-secure area of each bank (4-Kbyte page granularity) to be defined. The code executed in this HDP area, with its related data and keys, can be hidden after boot until the next system reset.

The hide protection principle is pictured in *Figure 8*.

Figure 8. Flash memory secure HDP area



When the HDPx (x=1,2) area is enabled and the corresponding HDPx_ACCDIS byte is different than its reset value 0xA3, data read, write and instruction fetch on the area defined by SECWMx_STRT and HDPx_END option bytes, are denied until the next device reset. The end of the HDPx areas can be extended (dynamically by the application) thanks to FLASH_SECHDPEXTR register. Refer to *Section 7.5.3: Secure hide protection (HDP)* and *Section 7.5.4: Secure hide protection extension (HDP extension)* for details.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

The HDP area can be resized by a secure application if the area is not hidden, and if RDP level ≠ 2.

3.6.2 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 3.5*, the hardware and software resources of STM32U3 series devices can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, available even if TrustZone is deactivated (TZEN = 0), critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions, each aligned to a multiple of 32 bytes. Each memory regions can be programmed to generate faults when accessed inappropriately by unprivileged software.

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.

For masters other than the Cortex-M33, the GTZC can assign them as unprivileged initiators, automatically protecting resources defined as privileged against this master.

Securing peripherals with TZSC (privileged-only)

In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:

- A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate that is controlled by the TZSC.
- A privileged-aware peripheral or memory is connected directly to the AHB or APB interconnect, implementing a specific behavior such as a subset of registers or a memory area is privilege-only.

When such peripheral is made privileged-only with GTZC, if it is master on the interconnect (SDMMC), it automatically issues privileged transactions. Privilege-aware masters like GPDMA, drive privileged signal in the AHB interconnect according to their internal privileged mode, independently to the GTZC.

The list of securable peripherals can be found in *Section 5: Global TrustZone controller (GTZC)*.

Securing memories with TZSC and MPCBB (privileged-only)

The GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 3.5.4*.

Error management (privileged-only)

- Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.
- The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write.
- When an illegal unprivileged access occurs, no bus error is generated, except when this access is an instruction fetch, accessing a privileged memory or a peripheral register.

Managing security in privileged-aware peripherals

TrustZone-aware peripherals also implement privileged-only access mode. The privileged protection is valid even if TZEN = 0:

Embedded flash memory

By default all embedded flash registers can be read or programmed in both privileged and unprivileged modes.

When secure privileged bit SPRIV is set in FLASH_PRIVCFGR, reading and writing the flash secure registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is deactivated (TZEN = 0).

When nonsecure privileged bit PRIV is set in FLASH_PRIVCFGR, reading and writing the flash nonsecure registers are possible only in privileged mode.

Regarding privileged protection of the embedded flash memory, the devices offer the following features:

- The system flash memory can be accessed both in privileged and unprivileged modes.
- Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.
- Each 4-Kbyte page of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIV1BBRx and FLASH_PRIV2BBRx. An unprivileged page is accessible by a privileged or unprivileged access.

Note: Switching a page from privileged to unprivileged does not erase the content of the page.

When applicable, an erase or program operation is always available to privileged code, and is available to unprivileged code only for unprivileged pages or unprivileged memory.

Direct memory access controllers (GPDMA)

When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules apply when accessing privileged/unprivileged source or destination. Those rules are summarized in *Table 12*.

Table 12. DMA channel use (privilege)

Destination	Privileged DMA channel x (PRIVx = 1)		Unprivileged DMA channel y (PRIVy = 0)	
	Privileged source	Unprivileged source	Privileged source	Unprivileged source
Privileged	OK		Transfer blocked(1)	
Unprivileged			Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return zeros.

See *Section 15: General purpose direct memory access controller (GPDMA)* for more details.

Power control (PWR)

By default, after a power-on or a system reset, all PWR registers but PWR_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

When secure privileged bit SPRIV is set in PWR_PRIVCFGR, reading and writing the PWR securable registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

When nonsecure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR nonsecure registers are possible only in privileged mode.

See *Section 9: Power control (PWR)* for details.

Secure clock and reset (RCC)

By default, after a power-on or a system reset, all RCC registers but RCC_PRIVCFGR can be read or written in both privileged and unprivileged modes.

When the secure privileged bit SPRIV is set in RCC_PRIVCFGR, reading and writing the RCC securable bits are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

When nonsecure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC nonsecure bits are possible only in privileged mode.

See *Section 10: Reset and clock control (RCC)* for details.

Real time clock (RTC)

By default after a backup domain reset, all RTC registers but RTC_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

When PRIV bit is set in privileged-only RTC_PRIVCFGR:

Writing the RTC registers is possible only in privileged mode.

Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privileged and unprivileged modes.

All the other RTC registers can be read only in privileged mode.

When PRIV bit is cleared in privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode to some RTC registers by setting dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV.

See *Section 43: Real-time clock (RTC)* chapter for details.

Tamper and backup registers (TAMP)

By default after any backup domain reset, all TAMP registers but TAMP_PRIVCFGR can be read or written in both privileged and unprivileged modes.

When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

- Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.
- Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counters that have their own protection setting.

The application can also:

- make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR
- increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPRIV bits in TAMP_PRIVCFGR:
  - Make protection zone 1 read privileged, write privileged.
  - Make protection zone 2 read privileged or unprivileged, write privileged.
  - Protection zone 3 is always read and write privileged or unprivileged.

General-purpose I/Os (GPIO)

All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

Extended interrupts and event controller (EXTI)

The EXTI peripheral is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in the privileged-only EXTI_PRIVCFGR1 register. When an input event is configured as privileged, only a privileged application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

The security configuration in EXTI_PRIVCFGR1 can be globally locked after reset in EXTI_LOCKR.

See *Section 17: Extended interrupts and event controller (EXTI)* for more details.

System configuration controller (SYSCFG)

All SYSCFG registers can be read and written in both privileged and unprivileged modes, except:

- FPUSEC bit in SYSCFG_SECCFGR registers (privileged only)
- SYSCFG registers for CPU configuration: SYSCFG_CSLOCKR, SYSCFG_FPUIMR and SYSCFG_CNSLCKR

See *Section 13: System configuration controller (SYSCFG)* for more details.

3.7 Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

3.7.1 Memory protection unit (MPU)

The Cortex-M33 includes a memory protection unit (MPU) that can restrict the read and write accesses to memory regions (including regions mapped to peripherals), based on one or more of the following parameters:

- Cortex-M33 operating mode (privileged, unprivileged)
- data/instruction fetch

The memory map and the programming of the nonsecure and secure MPUs split memory into regions (up to eight for the nonsecure, and up to twelve for the Trustzone). Secure MPU is only available when TrustZone is activated.

3.7.2 Embedded flash memory write protection

The embedded flash memory write protection (WRP) prevents illegal or unwanted write/erase to special sections of the embedded flash memory user area (system area is permanently write protected).

Write protected area is defined through the option bytes, writing the start and end addresses: two write-protected areas can be defined in each bank, with the granularity of a 4-Kbyte page.

WRP areas can be modified through option byte changes unless the corresponding FLASH_WRPxA/BR has its UNLOCK option bit cleared (meaning ROM emulation). UNLOCK can be set only when regressing from RDP level 1 to level 0.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

3.7.3 Tamper detection and response

Tamper detection sources

The devices support five input pins, allowing five static tamper inputs.

The tamper detection is functional in all system operating modes (Run, Sleep, Stop, Standby, or Shutdown), and in V_{BAT} mode.

Two modes are supported: edge detection, for which no clock is needed, and level detection with filtering. In this last case, the filtering is performed using either LSI or LSE clock (only LSE is available in Shutdown and V_{BAT} modes), and the tamper detection event is generated when either 2, 4 or 8 consecutive samples are observed at the selected level.

Note: Timestamps are automatically generated when a tamper event occurs.

The internal tamper sources are listed in *Table 446: TAMP interconnection*. Refer to *Section 44: Tamper and backup registers (TAMP)* for more details.

Response to tampers

Each source of tamper in the device can be configured to trigger the following events:

- Generate an interrupt, capable of waking up the device from Stop and Standby modes (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.

- Erase device secrets if the corresponding TAMPxPOM bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - symmetric keys stored in backup registers (x32), in SAES, AES, HASH, and CCB
  - asymmetric keys stored in PKA SRAM, erased when V_{DD} is present
  - other secrets stored in SRAM2 and CPU instruction cache memory (SRAM2 erased when V_{DD} is present)
  - nonvolatile information used to derive the DHUK in SAES is zeroed until complete SRAM2 erase
  - ICACHE erased when V_{DD} is present

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

Note: Device secret erase is also triggered by setting BKERASE in TAMP_CR2, or by performing an RDP regression as defined in Section 3.10.1.

Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxPOM bit in TAMP_CR2 (for external tamper pin) or TAMP_CR3 (for internal tamper).

In such situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- DHUK in SAES: fixed to a dummy value
- Backup registers, SRAM2: read-as-zero, write-ignored
- AES, SAES, CCB, and HASH peripherals: automatically reset by RCC
- PKA peripheral: reset, with memory use blocked (meaning PKA not usable)

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application just clears the flags to release secrets blocking (false tamper).

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset triggers automatically the erase of secrets.

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection are summarized in *Table 450: Effect of low-power modes on TAMP*.

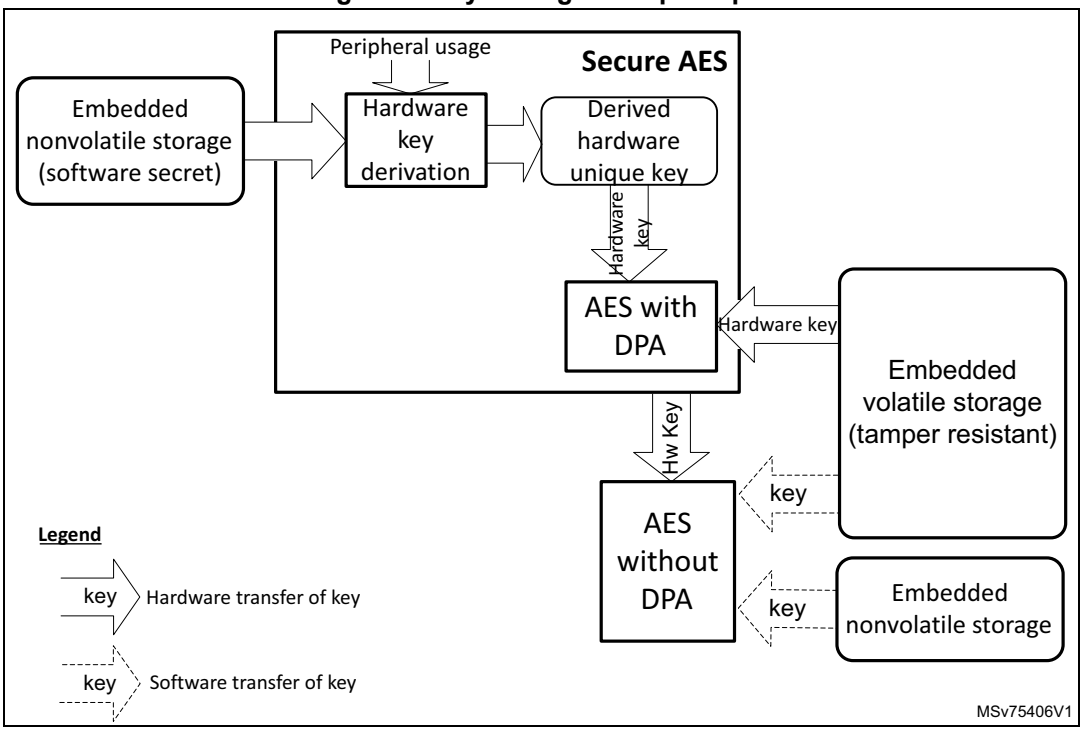
3.8 Secure storage

A critical feature of any security system is how long-term keys are stored, protected, and provisioned. Such keys are typically used for loading a boot image, or handling of critical user data.

Figure 9 shows how the key management service application can use the AES engine, for example, to compute external image decryption keys. A nonvolatile key can be stored in the embedded secure HDP area (see *Section 3.6.1*), while volatile key storage consists in the battery-powered, tamper-protected SRAM or registers in TrustZone-aware TAMP.

Figure 9 also shows keys that are manipulated by software, or keys that are managed only by hardware (like DHUK). More information on those hardware keys can be found in *Section 3.8.1*.

Figure 9. Key management principle



Tamper protection is detailed in *Section 3.7.3*, while TAMP TrustZone features are briefly described in *Section 3.5.5*.

3.8.1 Hardware secret key management

As shown in the previous figure, the devices propose a better protection for application keys, using hardware secret keys. These AES keys can be made usable to the application, without exposing them in clear-text (unencrypted). Such keys also become immediately unusable in case of tamper.

There are three different sources of hardware secret keys:

- DHUK: derived keys based on 256-bit nonvolatile device unique secret in flash memory, also known as Root Hardware Unique Key (RHUK). The generation of this key takes into account the TrustZone state and key use state (KMOD).

  Note: DHUK is the same for all devices when RDP = 0 (debug/development mode).
- BHK: 256-bit application key stored in tamper-resistant volatile storage in TAMP

  This key is written at boot time, then read/write locked to application until next reset.
- XORK: result of an XOR of BHK and DHUK

These keys can be used:

- as normal key, loading in write-only key registers (software key mode)
- as encryption/decryption key for another key, to be used in the DPA-resistant SAES (wrapped key mode)
- as encryption/decryption key for another key, to be used in a faster AES engine (shared key mode)

3.8.2 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 54.1: Unique device ID register (96 bits)*).

Application services can use this unique identity key to identify the product in the cloud network, or make it difficult for counterfeit devices or clones to inject untrusted data into the network.

Alternatively, the 256-bit device unique key (RHUK) can be used (see *Section 3.8.1*).

3.9 Cryptographic engines

The devices implement state-of-the-art cryptographic algorithms featuring key sizes and computing protection as recommended by national security agencies such as NIST for the U.S.A, BSI for Germany or ANSSI for France. Those algorithms are used to support privacy, authentication, integrity, entropy and identity attestation.

The cryptographic engines embedded in STM32 reduce the weaknesses on the implementation of critical cryptographic functions, preventing, for example, the use of weak cryptographic algorithms and key sizes. They also enable lower processing times and lower power consumption when performing cryptographic operations, offloading those computations from Cortex-M33. This is especially true for asymmetric cryptography.

For product certification purpose, ST can provide certified device information on how these security functions are implemented and validated.

For more information on cryptographic engine processing times, refer to their respective sections in the reference manual.

3.9.1 Cryptographic engines features

Table 13 lists the accelerated cryptographic operations available in the devices. Two AES accelerators are available (both can be reserved to secure application only).

Note: Additional operations can be added using the firmware.

The PKA can accelerate asymmetric cryptographic operations (like key pair generation, ECC scalar multiplication, point on curve check). See Section 34: Public key accelerator (PKA) for more details.

Table 13. Accelerated-cryptographic operations

Operations	Algorithm	Specification	Key lengths (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware modes running in parallel(2)
Encryption, decryption	AES(3)	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D	128, 256	GCM, CCM
Cipher-based message authentication code		NIST SP800-38D	128, 256	GMAC
Checksum	SHA-1	FIPS PUB 180-4	N/A	Digest 160-bit
Cryptographic hash	SHA-2			SHA-224, SHA-256, SHA-384, SHA-512
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (> 64 bytes)	-
Encryption/decryption key-pair generation(4)	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature(4) with hashing Signature verification	RSA	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	Refer to the table ‘Family of supported curves for ECC operations’ in *Section 34: Public key accelerator (PKA)* for details.
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.
2. Random numbers distribution to SAES and PKA using a dedicated hardware bus.
3. Protected against side-channel and timing attacks in SAES (see *Section 3.9.2*).
4. Private key cryptography protected against side-channel and timing attacks.

Note: Binary curves, Edwards curves, and Curve25519 are not supported by the PKA.

3.9.2 Secure AES co-processor (SAES)

The devices provide an additional on-chip hardware AES encryption and decryption engine, that implements counter-measures and mitigations against power and electromagnetic sidechannel attacks.

Clocked by the AHB bus clock and a dedicated kernel clock, SAES is slower than the AES, in order to provide best-in-class side-channel protections. The SAES engine supports 128-bit or 256bit key in electronic code book (ECB), cipher block chaining (CBC), (CTR), (GCM), (CCM), (GMAC) modes.

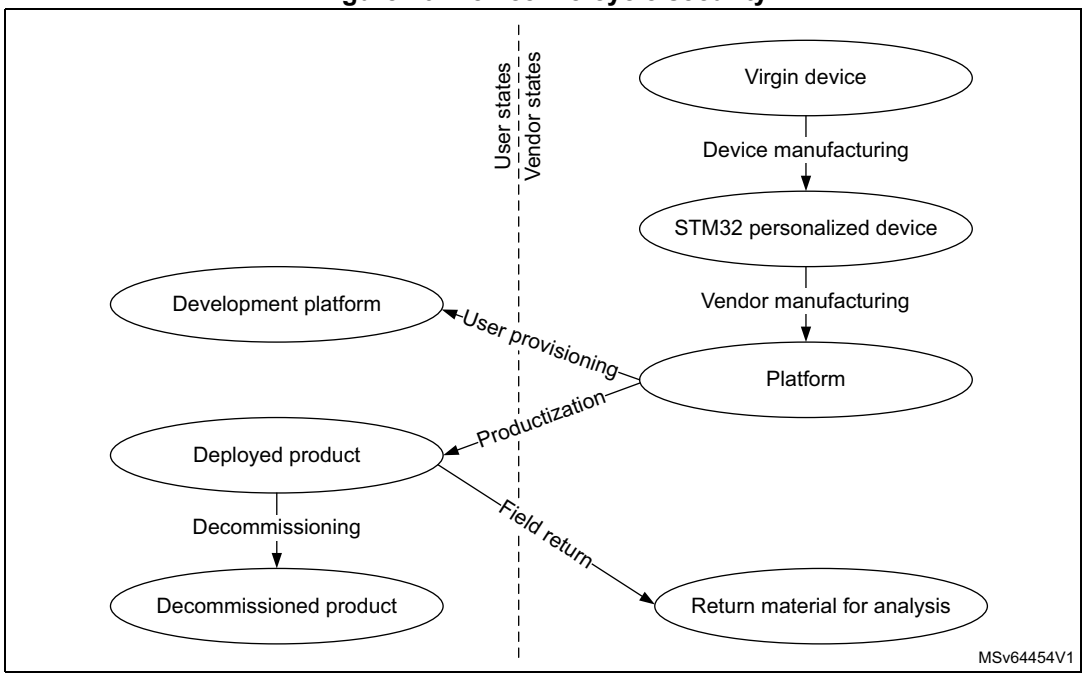
As shown in *Section 3.8*, the SAES can be used for extra-secure on-chip storage for sensitive information. It can also be made secure-only.

For more information, refer to *Section 32: Secure AES coprocessor (SAES)*.

3.10 Product life-cycle

A typical IoT device life-cycle is summarized in the figure below. For each step, the devices propose secure life-cycle management mechanisms embedded in the hardware.

Figure 10. Device life-cycle security



More details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in *Table 14*.

Table 14. Main product life-cycle transitions

Transitions	Description
Device manufacturing	STMicroelectronics creates new STM32 devices, always checking for manufacturing defects. During this process STM32 is provisioned with ROM firmware, secure firmware install (SFI) unique key pair, and a public ID.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“production” transition) or he/she can use the platform for software development (“user provisioning” transition).
Production	The end-user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return or decommissioning	These are one-way transitions, with devices kept in user premises or returned to the manufacturer. In both cases, all data including user data is destroyed, therefore the devices lose the ability to operate securely (like connecting to a managed IoT network).

The features described hereafter contribute to secure the device life-cycle.

3.10.1 Life-cycle management with readout protection (RDP)

The readout protection mechanism (full hardware feature) controls the access to the devices debug, test and provisioned secrets, as summarized in *Table 15*.

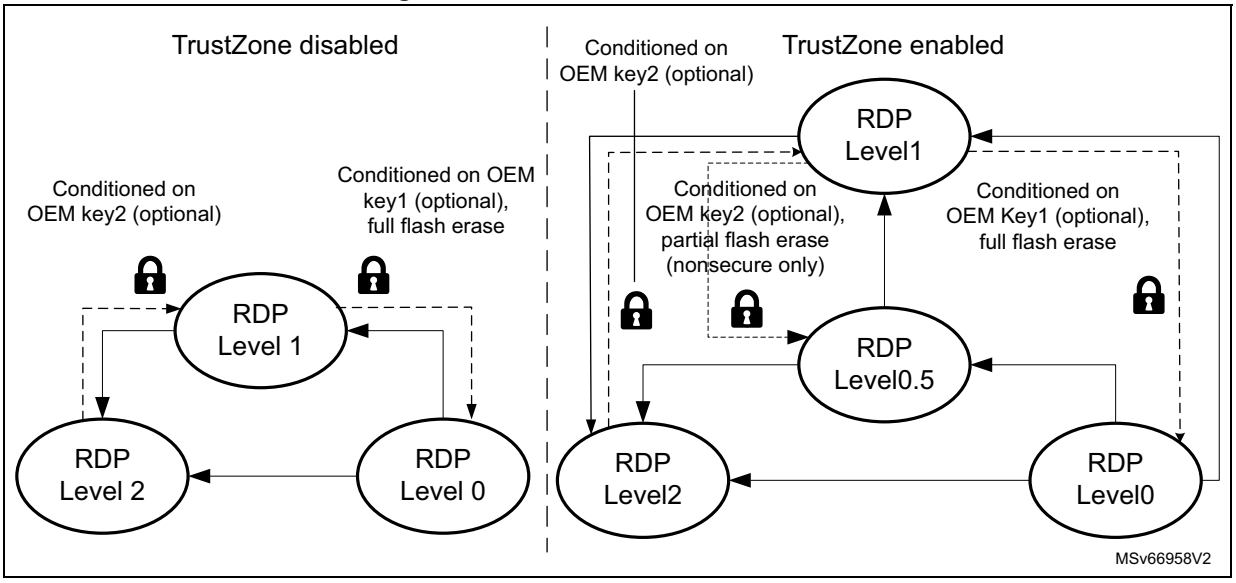
Table 15. Typical product life-cycle phases

RDP protection level		Debug	Comments
Level 0	Device open	Secure(1) and nonsecure	Boot address must target a secure area when TrustZone is enabled (secure SRAM, secure flash memory, RSS in system flash memory). Both OEM1 and OEM2 unlocking keys can be provisioned in the flash memory user options. The DHUK in the SAES peripheral is the same for all devices.
Level 0.5(2)	Device partially closed (closed-secure)	Nonsecure only	Boot address must target a secure area when TrustZone is enabled (secure user or system flash memory). Boot on SRAM is not permitted. Access to nonsecure flash memory is allowed when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the flash memory user options. The DHUK is different for every device.
Level 1	Device memories protected	Nonsecure only (conditioned)	Boot address must target the secure user flash memory. Accesses to nonsecure flash memory, encrypted flash memory, SRAM2, and backup registers are **not allowed** when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the flash memory user options. The DHUK is different for every device.
Level 2	Device closed	None (JTAG fuse)	Boot address must target the user flash memory (secure if TZEN = 1). Option bytes are read-only, hence RDP level 2 cannot be changed, unless the OEM2 unlocking key is activated (see *Table 16*). The DHUK is different for every device.

1. Debug is not available when executing RSS code.
2. Only applicable when TrustZone security is activated in the product.

The supported transitions, summarized in *Figure 11*, can be requested (when available) through the debug interface or via the system bootloader.

Figure 11. RDP level transition scheme



As shown in the previous figure, the user flash memory is automatically erased, either partially or in totality, during an RDP regression from RDP1. Those regressions can be conditioned to dedicated 128-bit password keys, if provisioned by the OEM (see next subsection). During the regression from RDP level 1 to RDP level 0.5, only nonsecure embedded flash memory is erased, keeping functional, for example, the secure boot and the secure firmware update. In all regressions from level 1, the OTP area in the flash memory is kept, all SRAMS and targeted device secrets are erased. Hence, no secrets must be stored in the OTP as they are revealed after a regression to RDP level 0. These secrets, also erased as response to tamper, are defined in *Section 3.7.3*.

Note: Enabling TrustZone using the option byte TZEN is only possible when RDP level is 0.

For more details on RDP, refer to *Section 7: Embedded flash memory (FLASH)*.

RDP unlocking sequences

The use of the two OEM password keys described in the last figure is further described hereafter.

Note: The devices support both permanent RDP level 2 (legacy mode) or password-based RDP level 2 regression to level 1. This level 2 regression does not erase the application code, and it does not change the RDP level 1 protections in place.

Details on the password-based regression can be found in *Table 16*.

Table 16. OEM1/2 RDP unlocking methods

OEM1 password options			OEM2 password options		
OEM1 LOCK	**Initial RDP level**	**RDP regression**	**OEM2 LOCK**	**Initial RDP level**	**RDP regression**
1	1	Regression to level 0 possible only through OEM1 unlock sequence (see below)	1	1	Regression to level 0.5 possible only through OEM2.1 unlock sequence (see below)
				2	Automatic regression to level 1 triggered upon successful OEM2.2 unlock sequence (see below)
0		Regression to level 0 always granted	0	1	Regression to level 0.5 always granted
				2	Regression to level 1 never granted RDP remains a permanent state.

Note: RDP regression with OEM key is described in section OEM1 RDP lock mechanism and OEM2 RDP lock mechanism.

JTAG 32-bit device specific ID

Unless the JTAG port is deactivated (OEM2LOCK = 0 and RDP level = 2), a 32-bit device specific quantity can always be read though the JTAG port. This information is stored in DBGMCU_DBG_AUTH_DEVICE.

The OEM can use this 32-bit information to derive the expected OEM password keys to unlock this specific device.

3.10.2 Recommended option-byte settings

Most of the time, the user threat model focuses mainly on software attacks. In this case, it may be sufficient to keep the RDP level 1 as device protection.

For a more aggressive threat model, where the user fears physical attacks on the STM32 device, it is recommended to optimize the level of security by setting the RDP level 2.

The recommended settings are detailed below:

- If TrustZone is disabled (TZEN = 0)
  - RDP level 2
  - nonsecure boot address option bytes set in user flash memory
- If TrustZone is enabled (TZEN = 1)
  - RDP level 2
  - BOOT_LOCK = 1
  - secure boot address option bytes set in user secure flash memory

As described in the previous section, the customer can decide to allow any RDP level 2 part to regress to RDP level 1, provided the OEM Key2 has been successfully provisioned, and the OEM2LOCK option bit is set.

3.11 Access controlled debug

The device restricts access to embedded debug features, in order to guarantee the confidentiality of customer assets against unauthorized usage of debug and trace features.

3.11.1 Debug protection with readout protection (RDP)

As described in *Section 3.10.1,* the hardware RDP mechanism automatically controls the accesses to the device debug and test. The protection of these debug features is defined in *Table 17*. Possible password-based regressions are described in *Section 3.10.1*.

Table 17. Debug protection with RDP

RDP protection level		Debug features protection
Level 0	Device open	Any debug(1)
Level 0.5(2)	Device partially closed	Secure debug is no longer available.
Level 1	Device memories protected	Nonsecure debug can no longer debug code and data stored in the embedded flash memory, the encrypted external flash memory, SRAM2, and backup registers.
Level 2	Device closed	JTAG is physically deactivated, unless it is kept operational only for password key injection (OEM2LOCK = 1). See *Section 3.10.1* for details.

1. Including ST engineering test modes, used for field returns.
2. Only applicable when TrustZone security is activated in the product.

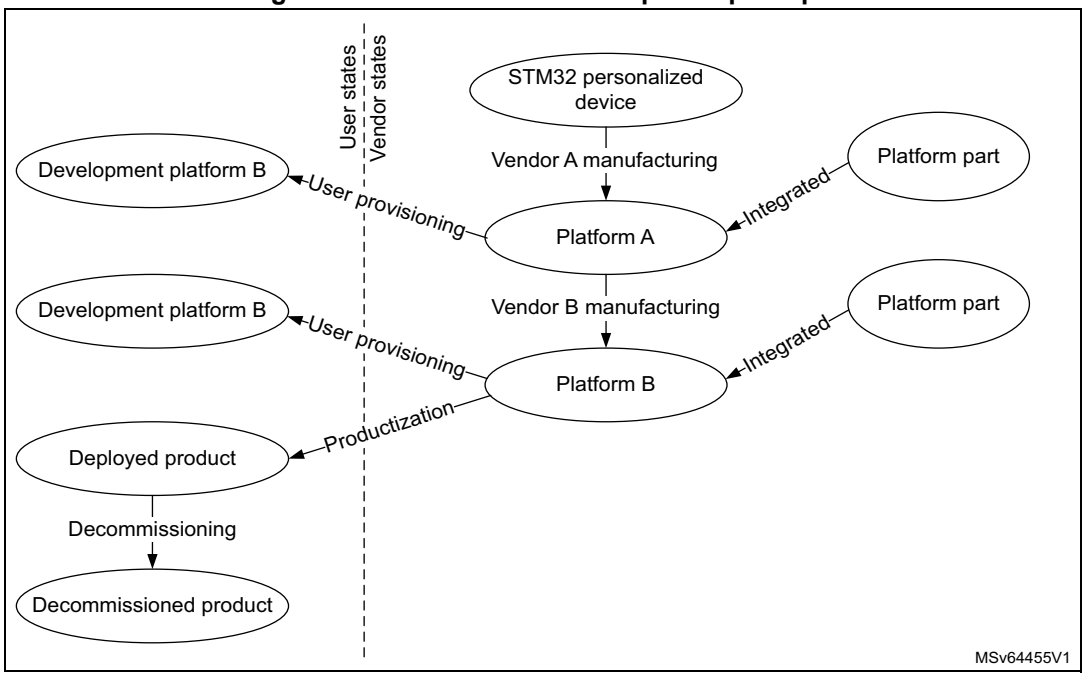
3.12 Software intellectual property protection and collaborative development

Thanks to the software intellectual property protection and collaborative model, the devices allow the design of solutions integrating innovative third-party libraries.

Collaborative development is summarized on the figure below. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, that can then be used by a vendor B, who does the same before deploying a final product to the end users.

Note: Each platform vendor can provision individual platforms for development not to be connected to a production cloud network (“Development Platform X”).

Figure 12. Collaborative development principle



The features described hereafter contribute to securing the software intellectual property within such a collaborative development.

3.12.1 Software intellectual property protection with RDP

As described in *Section 3.10.1,* the hardware RDP mechanism automatically controls the accesses to secrets provisioned in the device.

The protection of these secrets are defined in *Table 18*.

Table 18. Software intellectual property protection with RDP

RDP protection level		Secrets protection
Level 0	Device open	No special protections.
Level 0.5(1)	Device partially closed	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged, or traced
Level 1	Device memories protected	Data and code stored in embedded flash memory, SRAM2, and backup registers are no longer accessible through the debugger.
Level 2	Device closed	All data and code stored in the device cannot be dumped clear-text, debugged or traced.

1. Only applicable when TrustZone security is activated in the product.

3.12.2 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds.
- Non-invasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels such as timing, power and EM emissions.

4 Boot modes

At startup, NBOOT0 and NSWBOOT0 option bits of the FLASH_OPTR register, and ADD[24:0] option bytes of the FLASH_BOOT0R, FLASH_BOOT1R or FLASH_SBOOT0R registers are used to select the boot memory address that includes:

- Boot from any address in user flash memory
- Boot from system memory (bootloader)
- Boot from any address in the embedded SRAM
- Boot from root security service (RSS)

The BOOT0 value may come from the BOOT0 pin or from an option bit depending on the value of a user option bit to free the GPIO pad if needed.

The bootloader, located in the system memory, is used to reprogram the flash memory by using USART, I2C, I3C, SPI, FDCAN, or USB in device mode through the DFU (device firmware upgrade).

Table 19 and *Table 20* detail the boot modes when TrustZone is disabled or enabled.

Table 19. Boot modes when TrustZone is disabled (TZEN = 0)

NBOOT0	BOOT0 pin	NSW BOOT0	Boot address option-byte selection	Boot area	ST programmed default value
-	0	1	ADD[24:0] in FLASH_BOOT0R	Nonsecure boot base address 0 defined by user option bytes in FLASH_BOOT0R	Flash memory: 0x0800 0000
-	1	1	ADD[24:0] in FLASH_BOOT1R	Nonsecure boot base address 1 defined by user option bytes in FLASH_BOOT1R	Bootloader: 0x0BF8 F000
1	-	0	ADD[24:0] in FLASH_BOOT0R	Nonsecure boot base address 0 defined by user option bytes in FLASH_BOOT0R	Flash memory: 0x0800 0000
0	-	0	ADD[24:0] in FLASH_BOOT1R	Nonsecure boot base address 1 defined by user option bytes in FLASH_BOOT1R	Bootloader: 0x0BF8 F000

When TrustZone is enabled by setting the TZEN option bit, the boot space must be in the secure area. The ADD[24:0] option bytes in FLASH_SBOOT0R are used to select the boot secure memory address.

A unique boot entry option can be selected by setting the BOOT_LOCK option bit. All other boot options are ignored.

Table 20. Boot modes when TrustZone is enabled (TZEN = 1)

BOOT_ LOCK	NBOOT0	BOOT0 pin	NSW BOOT0	RSS command	Boot address option-byte selection	Boot area	ST programmed default value
0	-	0	1	0	ADD[24:0] in FLASH_SBOOT0R	Secure boot base address 0 defined by user option bytes in FLASH_SBOOT0R	Flash memory: 0x0C00 0000
	-	1	1	0	N/A	RSS	RSS: 0x0FF8 0000
	1	-	0	0	ADD[24:0] in FLASH_SBOOT0R	Secure boot base address 0 defined by user option bytes in FLASH_SBOOT0R	Flash memory: 0x0C00 0000
	0	-	0	0	N/A	RSS	RSS: 0x0FF8 0000
	-	-	-	≠ 0	N/A	RSS	RSS: 0x0FF8 0000
1	-	-	-	-	ADD[24:0] in FLASH_SBOOT0R	Secure boot base address 0 defined by user option bytes in FLASH_SBOOT0R	Flash memory: 0x0C00 0000

The boot address option bytes are used to program any boot memory address. However, the allowed address space depends on flash memory read protection RDP level.

If the programmed boot memory address is out of the allowed memory mapped area when RDP level is 0.5 or more, the default boot fetch address is forced either in the secure flash memory or the nonsecure flash memory depending on the TrustZone security option as described in the table below.

Table 21. Boot space versus RDP protection

RDP	TZEN = 1	TZEN = 0
0	Any boot address	Any boot address
0.5	Boot address only in RSS: 0x0FF80000 or in secure flash memory 0x0C00 0000-0x0C0F FFFF Otherwise the boot address is forced to 0x0FF8 0000.	N/A
1		Any boot address
2		Boot address only in flash memory 0x0800 0000-0x080F FFFF Otherwise boot address forced is 0x0800 0000.

The BOOT0 value (either coming from the pin or the option bit) is latched upon reset release. It is up to the user to set NBOOT0 or BOOT0 values to select the required boot mode.

The BOOT0 pin or user option bit (depending on NSWBOOT0 in FLASH_OPTR) is also resampled when exiting Standby mode. Consequently, the BOOT0 pin or user option bit

must be kept in the required boot mode configuration in Standby mode. After the startup delay, the selection of the boot area is done before releasing the processor reset.

BOOT0 GPIO is configured as follows:

- In input mode during the complete reset phase, if the option bit NSWBOOT0 is set in FLASH_OPTR, and then switches automatically in analog mode after reset is released (BOOT0 pin)
- In input mode from the reset phase to the completion of the option byte loading, if the bit NSWBOOT0 is cleared in FLASH_OPTR (BOOT0 value coming from the option bit), and then switches automatically to the analog mode even if the reset phase is not complete

4.1 Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. Refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).

4.2 Embedded root security services (RSS)

The embedded RSS are located in the secure information block, programmed by ST during production. The RSS, for example, enable the SFI (secure firmware installation), through the RSSe SFI (RSS extension firmware).

This feature allows the customer to produce the confidentiality of the firmware to be provisioned into the STM32, when production is sub-contracted to an untrusted third party.

The RSS are available on all devices, after enabling the TrustZone through the TZEN option bit. Refer to the application note *Overview secure firmware install (SFI)* (AN4992) for more details.

5 Global TrustZone controller (GTZC)

5.0.1 GTZC introduction

This section describes the global TrustZone controller (GTZC) block that contains the following subblocks:

- TZSC: TrustZone security controller

This subblock defines the secure/privileged state of slave peripherals. The TZSC informs some peripherals (such as RCC or GPIOs) about the secure status of each securable peripheral, by sharing with RCC and I/O logic.

- MPCBB: memory protection controller - block based

This subblock configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.

- TZIC: TrustZone illegal access controller

This subblock gathers all illegal access events in the system and generates a secure interrupt towards NVIC.

These subblocks are used to configure TrustZone system security in a product having bus agents with programmable-security and privileged attributes such as:

- on-chip RAM with programmable secure and/or privileged blocks (pages)
- AHB and APB peripherals with programmable security and/or privileged access
- off-chip memories with secure and/or privileged areas

5.0.2 GTZC main features

The GTZC main features are listed below:

- 3 independent 32-bit AHB interface for TZSC, TZIC and MPCBB
- TZIC accessible only with secure transactions
- Secure and nonsecure access supported for privileged and unprivileged part of TZSC and MPCBB
- Set of registers to define product security settings:
  - Secure/privileged blocks for internal SRAMs
  - Secure/privileged access mode for securable peripherals
  - Secure/privileged access mode for securable masters

GTZC TrustZone system architecture

The Armv8-M supports security per TrustZone-M model with isolation between:

- a secure world, where usually security sensitive applications are run and critical resources are located
- a nonsecure or public world (such as usual non secure operating system and user space)

The TrustZone architecture is extended beyond AHB and Armv8-M with:

- AHB/APB bridge used as secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards APB agents

- PPC (peripheral protection controller) used as secure gate to block or propagate secure/nonsecure and privileged/unprivileged transaction towards AHB agents
- TrustZone block-based MPC firewalls used as secure gate to filter secure/non secure, privileged/ unprivileged access towards internal SRAMs

AHB and APB peripherals can be categorized as:

- **privileged**: peripherals protected by AHB/APB firewall stub that is controlled from TZSC to define privilege properties
- **secure**: peripherals always protected by an AHB/APB firewall stub. These peripherals are always secure (such as TZIC)
- **securable**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **nonsecure and unprivileged**: peripherals connected directly to AHB/APB interconnect without any secure gate
- **TrustZone-aware**: peripherals connected directly to AHB or APB bus and implementing a specific TrustZone behavior (such as a subset of registers being secure). TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (such as Armv8-M core or DMA)

AHB securable masters can be configured in the TZSC to be secure/nonsecure and/or privileged/unprivileged.

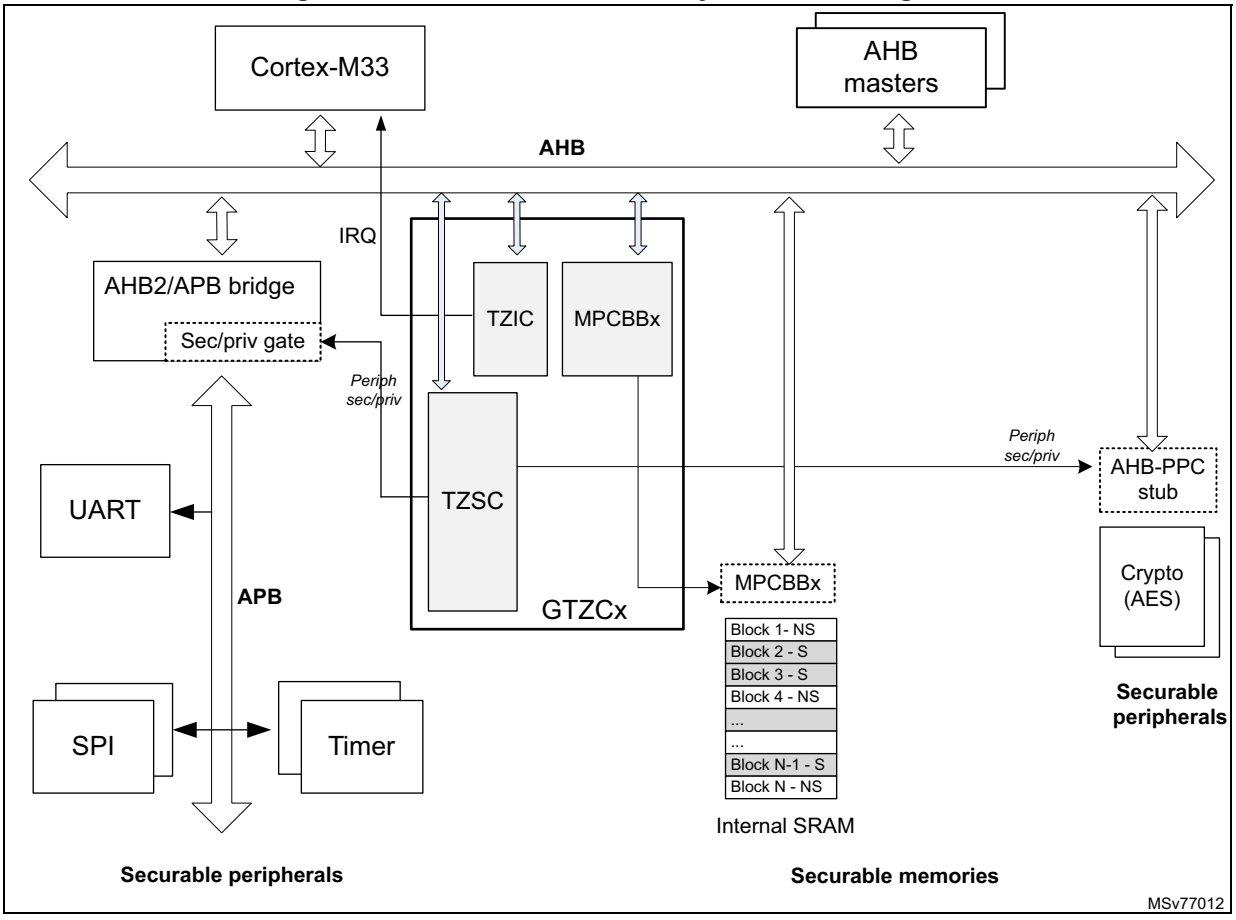
Application information

The TZSC, MPCBB and TZIC can be used in one of the following ways:

- programmed during secure boot only, locked and not changed afterwards
- dynamically re-programmed when using specific application code or secure kernel (microvisor). When not locked, MPC secure blocks or region size can be changed by secure software executing from the secure FLASH region or secure SRAM. Same remark applies to the GTZC_TZSC_SECCFGRx/PRIVCFGRx registers that define secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable and TrustZone-aware peripherals is shown in the figure below.

Figure 13. GTZC in Armv8-M subsystem block diagram



5.0.3 GTZC implementation

The STM32U3 series devices embed one instance of GTZC.

Table 22. GTZC features

GTZC subblocks	GTZC1
TZSC	X
TZIC	X
MPCBB subblock	MPCBB1/2

The table below shows the address offset of GTZC subblocks versus GTZC base address (refer to *Section 2.3.2* for GTZC1 base addresses).

Table 23. GTZC1 subblocks address offset

GTZC1 subblock	Address offset
GTZC1_TZSC	0x0
GTZC1_TZIC	0x400
GTZC1_MPCBB1	0x800
GTZC1_MPCBB2	0xC00

The table below describes the characteristics of the available MPCBB.

Table 24. MPCBB resource assignment

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	192	512	384	12
	MPCBB2	SRAM2	64		128	4

5.1 GTZC functional description

5.1.1 GTZC block diagram

Figure 14 describes the combined feature of TZSC, MPCBB and TZIC. Each subblock is controlled by its own AHB configuration port.

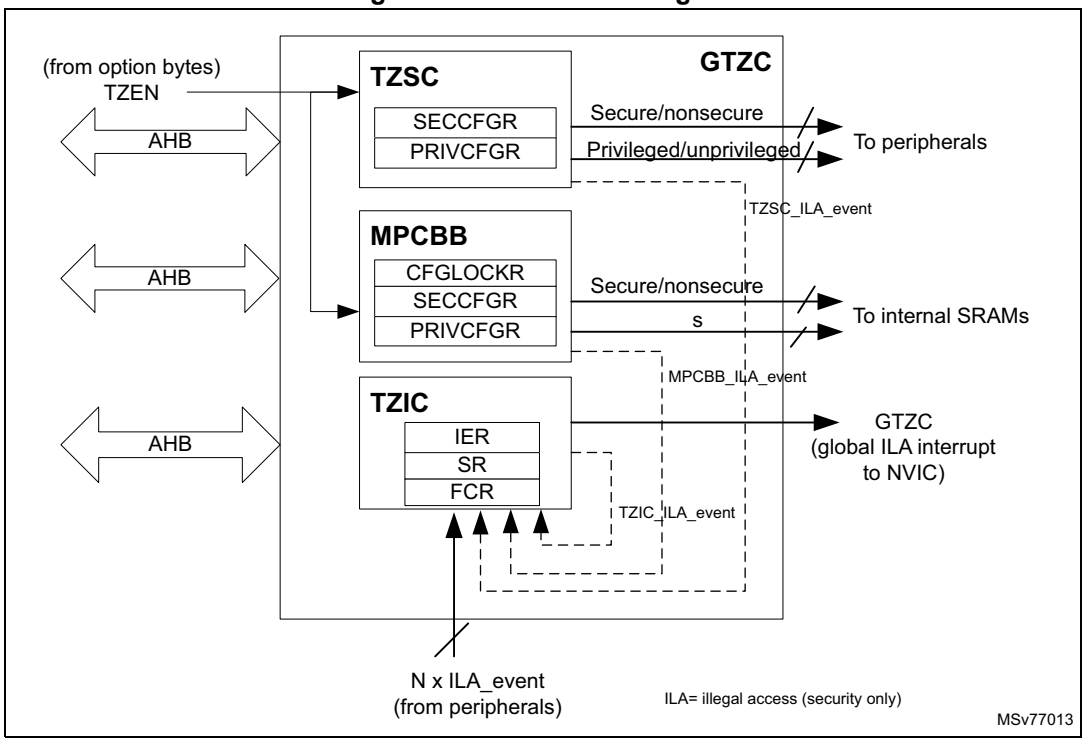
The TZSC defines which peripheral is secure and/or privileged.The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (nonsecure) is sufficient.

On the opposite, the secure configuration bit of a peripheral can be modified only with a secure privileged transaction if the peripheral is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The secure configuration bit of a given ram block can be modified only with a secure privileged transaction if the same RAM block is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The TZIC gathers illegal events generated within the system when an illegal access is detected. TZIC can then generate a secure interrupt towards the CPU if needed.

Figure 14. GTZC block diagram



5.1.2 Illegal access definition

Three different types of illegal access exist:

- Illegal nonsecure access

  Any nonsecure transaction trying to write a secure resource is considered as illegal and thus the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. However some exceptions exist on secure and privileged configuration registers: these later ones authorize non secure read access to secure registers (see GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx).

- Illegal secure access

  Any secure transaction trying to access nonsecure block in internal block-based SRAM is considered as illegal.

  Correct TZIC settings allows the capture of the associated event and then generates the GTZC_IRQn interrupt to the NVIC. This applies for read, write and execute access.

  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on nonsecure SRAM blocks, by setting the SRWILADIS bit in the GTZC_MPCBBz_CR register. Secure read and write data transactions are then allowed on nonsecure SRAM blocks, while secure execution access remains not allowed.

  Any secure execute transaction trying to access a nonsecure peripheral register is considered as illegal and generate a bus error.

- Illegal unprivileged access

  Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal read and write access. The

addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write. No bus error is generated. A bus error is generated when any unprivileged execute transaction tries to access a privileged memory.

5.1.3 TrustZone security controller (TZSC)

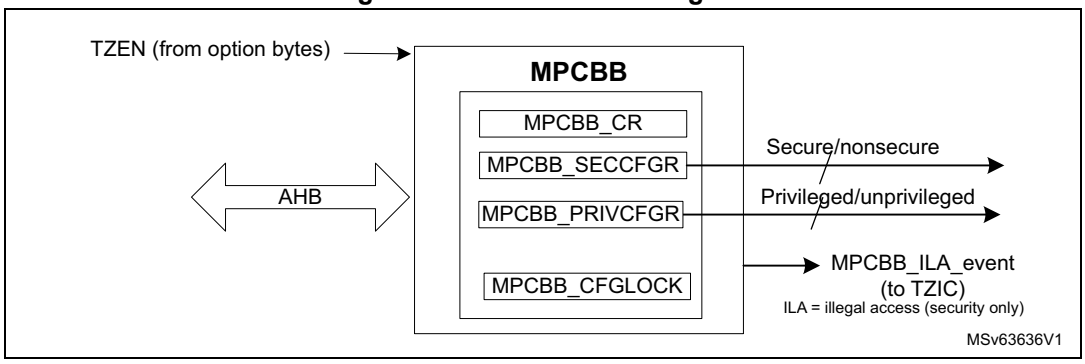
The TZSC is composed of a configurable set of registers, providing the following features:

- Control of secure and privileged state for all peripherals, done through:
  - GTZC_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals
  - GTZC_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals

5.1.4 Memory protection controller - block based (MPCBB)

The MPCBB is composed of a configurable set of registers allowing to define security and privileged policy for internal SRAM memories. The security and privileged policy can be individually configured per each 512-byte block of SRAM.

Figure 15. MPCBB block diagram



In order to setup the MPCBB, the following actions are needed (for example at boot time):

- Secure firmware must define which memory blocks are secure by setting the correct bits in GTZC_MPCBBz_SECCFGRx.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC_MPCBBz_PRIVCFGRx.

A MPCBB super-block is made of 32 consecutive blocks. For each super-block, secure application can lock all related security/privileged bits using the correct bits in GTZC_MPCBBz_CFGLOCKR1. This lock remains active until the next system reset.

*Note: The block size is 512 bytes.The super-block size is 512 * 32 = 16 Kbytes.*

5.1.5 TrustZone illegal access controller (TZIC)

The TZIC concentrates all illegal access source events. It is used only when the system is TrustZone enabled (TZEN = 1).

TZIC allows the trace (flag) of which event trigged the secure illegal access interrupt. Register masks (GTZC_TZIC_IERx) are available to filter unwanted event. On unmasked illegal event, TZIC generates the GTZC_IRQn interrupt to the NVIC.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC_TZIC_SRx and GTZC_TZIC_FCRx). The reset value of mask registers (GTZC_TZIC_IERx) is such that all events are masked.

5.1.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx, meaning that all securable peripherals are respectively set to nonsecure and unprivileged.

For internal SRAMx (x = 1 to 2), all GTZC_MPCBBz_SECCFGRx and GTZC_MPCBBz_PRIVCFGRx are set:

- to 0xFFFF FFFF, making these internal memories block secure and privileged by default when TrustZone security is enabled at system level (TZEN = 1).
- to 0x0000 0000, making these internal memories block nonsecure and unprivileged by default when TrustZone security is disabled at system level (TZEN = 0)

Secure boot code can then program the security settings, making components secure or not as needed.

5.1.7 GTZC interrupts

TZIC is a secure peripheral, thus it systematically generates an illegal access event when accessed by a nonsecure access. The MPCBB and TZSC are TrustZone-aware peripherals, meaning that secure and nonsecure registers co-exist within the peripheral.

Table 25. GTZC interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
GTZC	Illegal access	All flags in GTZC_TZIC_SRx	All bits in GTZC_TZIC_IERx	Write 1 in the bit GTZC_TZIC_FCRx	Yes	Yes	No

5.2 GTZC1 TZSC registers

All registers are accessed only by words (32-bit).

5.2.1 GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LCK**: Lock GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx configuration until next reset

This bit is cleared by default and once set, it cannot be reset until system reset.

0: Configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx registers not locked

1: Configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx registers locked

5.2.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register bit is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFSEC	OPAMPSEC	I3C1SEC	SPI3SEC	Res.	Res.	FDCAN1SEC	LPTIM2SEC	Res.
							rw	rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSSEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	Res.	SPI2SEC	IWDGSEC	WWDGSEC	TIM7SEC	TIM6SEC	Res.	TIM4SEC	TIM3SEC	TIM2SEC
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VREFBUFSEC**: Secure access mode for VREFBUF

0: Nonsecure

1: Secure

Bit 23 **OPAMPSEC**: Secure access mode for OPAMP

0: Nonsecure

1: Secure

Bit 22 **I3C1SEC**: Secure access mode for I3C1

0: Nonsecure

1: Secure

Bit 21 **SPI3SEC**: Secure access mode for SPI3

0: Nonsecure

1: Secure

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **FDCAN1SEC**: Secure access mode for FDCAN1

0: Nonsecure

1: Secure

Bit 17 **LPTIM2SEC**: Secure access mode for LPTIM2
0: Nonsecure
1: Secure

Bit 16 Reserved, must be kept at reset value.

Bit 15 **CRSSEC**: Secure access mode for CRS
0: Nonsecure
1: Secure

Bit 14 **I2C2SEC**: Secure access mode for I2C2
0: Nonsecure
1: Secure

Bit 13 **I2C1SEC**: Secure access mode for I2C1
0: Nonsecure
1: Secure

Bit 12 **UART5SEC**: Secure access mode for UART5
0: Nonsecure
1: Secure

Bit 11 **UART4SEC**: Secure access mode for UART4
0: Nonsecure
1: Secure

Bit 10 **USART3SEC**: Secure access mode for USART3
0: Nonsecure
1: Secure

Bit 9 Reserved, must be kept at reset value.

Bit 8 **SPI2SEC**: Secure access mode for SPI2
0: Nonsecure
1: Secure

Bit 7 **IWDGSEC**: Secure access mode for IWDG
0: Nonsecure
1: Secure

Bit 6 **WWDGSEC**: Secure access mode for WWDG
0: Nonsecure
1: Secure

Bit 5 **TIM7SEC**: Secure access mode for TIM7
0: Nonsecure
1: Secure

Bit 4 **TIM6SEC**: Secure access mode for TIM6
0: Nonsecure
1: Secure

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM4SEC**: Secure access mode for TIM4
0: Nonsecure
1: Secure

Bit 1 **TIM3SEC**: Secure access mode for TIM3
0: Nonsecure
1: Secure

Bit 0 **TIM2SEC**: Secure access mode for TIM2
0: Nonsecure
1: Secure

5.2.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register bit is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPSEC
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM4SEC	LPTIM3SEC	LPTIM1SEC	I2C3SEC	LPUART1SEC	I3C2SEC	USBSEC	Res.	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	Res.	SPI1SEC	TIM1SEC
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **COMPSEC**: Secure access mode for COMP
0: Nonsecure
1: Secure

Bit 15 **LPTIM4SEC**: Secure access mode for LPTIM4
0: Nonsecure
1: Secure

Bit 14 **LPTIM3SEC**: Secure access mode for LPTIM3
0: Nonsecure
1: Secure

Bit 13 **LPTIM1SEC**: Secure access mode for LPTIM1
0: Nonsecure
1: Secure

Bit 12 **I2C3SEC**: Secure access mode for I2C3
0: Nonsecure
1: Secure

Bit 11 **LPUART1SEC**: Secure access mode for LPUART1
0: Nonsecure
1: Secure

Bit 10 **I3C2SEC**: Secure access mode for I3C2
0: Nonsecure
1: Secure

Bit 9 **USBSEC**: Secure access mode for USB
0: Nonsecure
1: Secure

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SAI1SEC**: Secure access mode for SAI1
0: Nonsecure
1: Secure

Bit 6 **TIM17SEC**: Secure access mode for TIM17
0: Nonsecure
1: Secure

Bit 5 **TIM16SEC**: Secure access mode for TIM16
0: Nonsecure
1: Secure

Bit 4 **TIM15SEC**: Secure access mode for TIM15
0: Nonsecure
1: Secure

Bit 3 **USART1SEC**: Secure access mode for USART1
0: Nonsecure
1: Secure

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SPI1SEC**: Secure access mode for SPI1
0: Nonsecure
1: Secure

Bit 0 **TIM1SEC**: Secure access mode for TIM1
0: Nonsecure
1: Secure

5.2.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register bit is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1SEC	DAC1SEC	RAMCFGSEC	Res.	OCTOSPI1_REGSEC	Res.	Res.	SDMMC1SEC	CCBSEC
							rw	rw	rw		rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESSEC	PKASEC	RNGSEC	HASHSEC	AESSEC	Res.	Res.	ADC12SEC	Res.	ICACHE_REGSEC	Res.	TSCSEC	CRCSEC	Res.	Res.	Res.
rw	rw	rw	rw	rw			rw		rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADF1SEC**: Secure access mode for ADF1
- 0: Nonsecure
- 1: Secure

Bit 23 **DAC1SEC**: Secure access mode for DAC1
- 0: Nonsecure
- 1: Secure

Bit 22 **RAMCFGSEC**: Secure access mode for RAMCFG
- 0: Nonsecure
- 1: Secure

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1_REGSEC**: Secure access mode for OCTOSPI1 registers
- 0: Nonsecure
- 1: Secure

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC1SEC**: Secure access mode for SDMMC1
- 0: Nonsecure
- 1: Secure

Bit 16 **CCBSEC**: Secure access mode for CCB
- 0: Nonsecure
- 1: Secure

Bit 15 **SAESSEC**: Secure access mode for SAES
- 0: Nonsecure
- 1: Secure

Bit 14 **PKASEC**: Secure access mode for PKA
0: Nonsecure
1: Secure

Bit 13 **RNGSEC**: Secure access mode for RNG
0: Nonsecure
1: Secure

Bit 12 **HASHSEC**: Secure access mode for HASH
0: Nonsecure
1: Secure

Bit 11 **AESSEC**: Secure access mode for AES
0: Nonsecure
1: Secure

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADC12SEC**: Secure access mode for ADC1 and ADC2
0: Nonsecure
1: Secure

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGSEC**: Secure access mode for ICACHE registers
0: Nonsecure
1: Secure

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCSEC**: Secure access mode for TSC
0: Nonsecure
1: Secure

Bit 3 **CRCSEC**: Secure access mode for CRC
0: Nonsecure
1: Secure

Bits 2:0 Reserved, must be kept at reset value.

5.2.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_TZSC_SECCFGR register bit is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFPRIV	OPAMPPRIV	I3C1PRIV	SPI3PRIV	Res.	Res.	FDCAN1PRIV	LPTIM2PRIV	Res.
							rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSPRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	Res.	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM7PRIV	TIM6PRIV	Res.	TIM4PRIV	TIM3PRIV	TIM2PRIV
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VREFBUFPRIV**: Privileged access mode for VREFBUF
0: Unprivileged
1: Privileged

Bit 23 **OPAMPPRIV**: Privileged access mode for OPAMP
0: Unprivileged
1: Privileged

Bit 22 **I3C1PRIV**: Privileged access mode for I3C1
0: Unprivileged
1: Privileged

Bit 21 **SPI3PRIV**: Privileged access mode for SPI3
0: Unprivileged
1: Privileged

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **FDCAN1PRIV**: Privileged access mode for FDCAN1
0: Unprivileged
1: Privileged

Bit 17 **LPTIM2PRIV**: Privileged access mode for LPTIM2
0: Unprivileged
1: Privileged

Bit 16 Reserved, must be kept at reset value.

Bit 15 **CRSPRIV**: Privileged access mode for CRS
0: Unprivileged
1: Privileged

Bit 14 **I2C2PRIV**: Privileged access mode for I2C2
0: Unprivileged
1: Privileged

Bit 13 **I2C1PRIV**: Privileged access mode for I2C1
0: Unprivileged
1: Privileged

Bit 12 **UART5PRIV**: Privileged access mode for UART5
0: Unprivileged
1: Privileged

Bit 11 **UART4PRIV**: Privileged access mode for UART4
0: Unprivileged
1: Privileged

Bit 10 **USART3PRIV**: Privileged access mode for USART3
0: Unprivileged
1: Privileged

Bit 9 Reserved, must be kept at reset value.

Bit 8 **SPI2PRIV**: Privileged access mode for SPI2
0: Unprivileged
1: Privileged

Bit 7 **IWDGPRIV**: Privileged access mode for IWDG
0: Unprivileged
1: Privileged

Bit 6 **WWDGPRIV**: Privileged access mode for WWDG
0: Unprivileged
1: Privileged

Bit 5 **TIM7PRIV**: Privileged access mode for TIM7
0: Unprivileged
1: Privileged

Bit 4 **TIM6PRIV**: Privileged access mode for TIM6
0: Unprivileged
1: Privileged

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM4PRIV**: Privileged access mode for TIM4
0: Unprivileged
1: Privileged

Bit 1 **TIM3PRIV**: Privileged access mode for TIM3
0: Unprivileged
1: Privileged

Bit 0 **TIM2PRIV**: Privileged access mode for TIM2
0: Unprivileged
1: Privileged

5.2.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_TZSC_SECCFGR register bit is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPPRIV
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM4PRIV	LPTIM3PRIV	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	I3C2PRIV	USBPRIV	Res.	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **COMPPRIV**: Privileged access mode for COMP
0: Unprivileged
1: Privileged

Bit 15 **LPTIM4PRIV**: Privileged access mode for LPTIM4
0: Unprivileged
1: Privileged

Bit 14 **LPTIM3PRIV**: Privileged access mode for LPTIM3
0: Unprivileged
1: Privileged

Bit 13 **LPTIM1PRIV**: Privileged access mode for LPTIM1
0: Unprivileged
1: Privileged

Bit 12 **I2C3PRIV**: Privileged access mode for I2C3
0: Unprivileged
1: Privileged

Bit 11 **LPUART1PRIV**: Privileged access mode for LPUART1
0: Unprivileged
1: Privileged

Bit 10 **I3C2PRIV**: Privileged access mode for I3C2
0: Unprivileged
1: Privileged

Bit 9 **USBPRIV**: Privileged access mode for USB
0: Unprivileged
1: Privileged

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SAI1PRIV**: Privileged access mode for SAI1
0: Unprivileged
1: Privileged

Bit 6 **TIM17PRIV**: Privileged access mode for TIM17
0: Unprivileged
1: Privileged

Bit 5 **TIM16PRIV**: Privileged access mode for TIM16
0: Unprivileged
1: Privileged

Bit 4 **TIM15PRIV**: Privileged access mode for TIM15
0: Unprivileged
1: Privileged

Bit 3 **USART1PRIV**: Privileged access mode for USART1
0: Unprivileged
1: Privileged

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SPI1PRIV**: Privileged access mode for SPI1
0: Unprivileged
1: Privileged

Bit 0 **TIM1PRIV**: Privileged access mode for TIM1
0: Unprivileged
1: Privileged

5.2.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x0028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_TZSC_SECCFGR register bit is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by nonsecure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1PRIV	DAC1PRIV	RAMCFGPRIV	Res.	OCTOSPI1_REGPRIV	Res.	Res.	SDMMC1PRIV	CCBPRIV
							rw	rw	rw		rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESPRIV	PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	ADC12PRIV	Res.	ICACHE_REGPRIV	Res.	TSCPRIV	CRCPRIV	Res.	Res.	Res.
rw	rw	rw	rw	rw			rw		rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADF1PRIV**: Privileged access mode for ADF1
0: Unprivileged
1: Privileged

Bit 23 **DAC1PRIV**: Privileged access mode for DAC1
0: Unprivileged
1: Privileged

Bit 22 **RAMCFGPRIV**: Privileged access mode for RAMCFG
0: Unprivileged
1: Privileged

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1_REGPRIV**: Privileged access mode for OCTOSPI1 registers
0: Unprivileged
1: Privileged

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC1PRIV**: Privileged access mode for SDMMC1
0: Unprivileged
1: Privileged

Bit 16 **CCBPRIV**: Privileged access mode for CCB
0: Unprivileged
1: Privileged

Bit 15 **SAESPRIV**: Privileged access mode for SAES
0: Unprivileged
1: Privileged

Bit 14 **PKAPRIV**: Privileged access mode for PKA
0: Unprivileged
1: Privileged

Bit 13 **RNGPRIV**: Privileged access mode for RNG
0: Unprivileged
1: Privileged

Bit 12 **HASHPRIV**: Privileged access mode for HASH
0: Unprivileged
1: Privileged

Bit 11 **AESPRIV**: Privileged access mode for AES
0: Unprivileged
1: Privileged

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADC12PRIV**: Privileged access mode for ADC1 and ADC2
0: Unprivileged
1: Privileged

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGPRIV**: Privileged access mode for ICACHE registers
0: Unprivileged
1: Privileged

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCPRIV**: Privileged access mode for TSC
0: Unprivileged
1: Privileged

Bit 3 **CRCPRIV**: Privileged access mode for CRC
0: Unprivileged
1: Privileged

Bits 2:0 Reserved, must be kept at reset value.

5.2.8 GTZC1 TZSC register map

Table 26. GTZC1 TZSC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZSC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value																																0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC1_TZSC_SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFSEC	OPAMPSEC	I3C1SEC	SPI3SEC	Res.	Res.	FDCAN1SEC	LPTIM2SEC	Res.	CRSSEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	Res.	SPI2SEC	IWDGSEC	WWDGSEC	TIM7SEC	TIM6SEC	Res.	TIM4SEC	TIM3SEC	TIM2SEC
	Reset value								0	0	0	0			0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0
0x014	GTZC1_TZSC_SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPSEC	LPTIM4SEC	LPTIM3SEC	LPTIM1SEC	I2C3SEC	LPUARTSEC	I3C2SEC	USBSEC	Res.	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	Res.	SPI1SEC	TIM1SEC
	Reset value																0	0	0	0	0	0	0	0		0	0	0	0	0		0	0
0x018	GTZC1_TZSC_SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1SEC	DAC1SEC	RAMCFGSEC	Res.	OCTOSPI1_REGSEC	Res.	Res.	SDMMC1SEC	CCBSEC	SAESSEC	PKASEC	RNGSEC	HASHSEC	AESSEC	Res.	Res.	ADC12SEC	Res.	ICACHE_REGSEC	Res.	TSCSEC	CRCSEC	Res.	Res.	Res.
	Reset value								0	0	0		0			0	0	0	0	0	0	0			0		0		0	0			
0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFPRIV	OPAMPPRIV	I3C1PRIV	SPI3PRIV	Res.	Res.	FDCAN1PRIV	LPTIM2PRIV	Res.	CRSPRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	Res.	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM7PRIV	TIM6PRIV	Res.	TIM4PRIV	TIM3PRIV	TIM2PRIV
	Reset value									0	0	0			0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0
0x024	GTZC1_TZSC_PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPPRIV	LPTIM4PRIV	LPTIM3PRIV	LPTIM1PRIV	I2C3PRIV	LPUARTPRIV	I3C2PRIV	USBPRIV	Res.	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV
	Reset value																0	0	0	0	0	0	0	0		0	0	0	0	0		0	0

Table 26. GTZC1 TZSC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	GTZC1_TZSC_PRIVFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1PRIV	DAC1PRIV	RAMCFGPRIV	Res.	OCTOSPI1_REGPRIV	Res.	Res.	SDMMC1PRIV	CCBPRIV	SAESPRIV	PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	ADC12PRIV	Res.	ICACHE_REGPRIV	Res.	TSCPRIV	CRCPRIV	Res.	Res.	Res.
	Reset value								0	0	0		0			0	0	0	0	0	0	0			0		0		0	0			
0x02C-0x03C	Reserved	Reserved																															

Refer to *Section 5.0.3: GTZC implementation*.

5.3 GTZC1 TZIC registers

All registers are accessed only by words (32-bit).

5.3.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only. This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFIE	OPAMPIE	I3C1IE	SPI3IE	Res.	Res.	FDCAN1IE	LPTIM2IE	Res.
							rw	rw	rw	rw			rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSIE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	Res.	SPI2IE	IWDGIE	WWDGIE	TIM7IE	TIM6IE	Res.	TIM4IE	TIM3IE	TIM2IE
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VREFBUFIE**: Illegal access interrupt enable for VREFBUF
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 23 **OPAMPIE**: Illegal access interrupt enable for OPAMP
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 22 **I3C1IE**: Illegal access interrupt enable for I3C1
- 0: Interrupt disabled
- 1: Interrupt enabled

Bit 21 **SPI3IE**: Illegal access interrupt enable for SPI3
- 0: Interrupt disabled
- 1: Interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **FDCAN1IE**: Illegal access interrupt enable for FDCAN1 and FDCAN SRAM
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **LPTIM2IE**: Illegal access interrupt enable for LPTIM2
0: Interrupt disabled
1: Interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **CRSIE**: Illegal access interrupt enable for CRS
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **I2C2IE**: Illegal access interrupt enable for I2C2
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **I2C1IE**: Illegal access interrupt enable for I2C1
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **UART5IE**: Illegal access interrupt enable for UART5
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **UART4IE**: Illegal access interrupt enable for UART4
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **USART3IE**: Illegal access interrupt enable for USART3
0: Interrupt disabled
1: Interrupt enabled

Bit 9 Reserved, must be kept at reset value.

Bit 8 **SPI2IE**: Illegal access interrupt enable for SPI2
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **IWDGIE**: Illegal access interrupt enable for IWDG
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **WWDGIE**: Illegal access interrupt enable for WWDG
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **TIM7IE**: Illegal access interrupt enable for TIM7
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **TIM6IE**: Illegal access interrupt enable for TIM6
0: Interrupt disabled
1: Interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM4IE**: Illegal access interrupt enable for TIM4
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **TIM3IE**: Illegal access interrupt enable for TIM3
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **TIM2IE**: Illegal access interrupt enable for TIM2
0: Interrupt disabled
1: Interrupt enabled

5.3.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged access only. This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPIE
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C3IE	LPUART1IE	I3C2IE	USBIE	Res.	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	Res.	SPI1IE	TIM1IE
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **COMPIE**: Illegal access interrupt enable for COMP
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **LPTIM4IE**: Illegal access interrupt enable for LPTIM4
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **LPTIM3IE**: Illegal access interrupt enable for LPTIM3
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **LPTIM1IE**: Illegal access interrupt enable for LPTIM1
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **I2C3IE**: Illegal access interrupt enable for I2C3
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **LPUART1IE**: Illegal access interrupt enable for LPUART1
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **I3C2IE**: Illegal access interrupt enable for I3C2
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **USBIE**: Illegal access interrupt enable for USB
0: Interrupt disabled
1: Interrupt enabled

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SAI1IE**: Illegal access interrupt enable for SAI1
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **TIM17IE**: Illegal access interrupt enable for TIM17
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **TIM16IE**: Illegal access interrupt enable for TIM16
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **TIM15IE**: Illegal access interrupt enable for TIM15
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **USART1IE**: Illegal access interrupt enable for USART1
0: Interrupt disabled
1: Interrupt enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SPI1IE**: Illegal access interrupt enable for SPI1
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **TIM1IE**: Illegal access interrupt enable for TIM1
0: Interrupt disabled
1: Interrupt enabled

5.3.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged access only. This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1IE	DAC1IE	RAMCFGIE	Res.	OCTOSPI1_REGIE	Res.	Res.	SDMMC1IE	CCBIE
							rw	rw	rw		rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESIE	PKAIE	RNGIE	HASHIE	AESIE	Res.	Res.	ADC12IE	Res.	ICACHE_REGIE	Res.	TSCIE	CRCIE	Res.	Res.	Res.
rw	rw	rw	rw	rw			rw		rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADF1IE**: Illegal access interrupt enable for ADF1
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **DAC1IE**: Illegal access interrupt enable for DAC1
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **RAMCFGIE**: Illegal access interrupt enable for RAMCFG
0: Interrupt disabled
1: Interrupt enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1_REGIE**: Illegal access interrupt enable for OCTOSPI1 registers
0: Interrupt disabled
1: Interrupt enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC1IE**: Illegal access interrupt enable for SDMMC1
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **CCBIE**: Illegal access interrupt enable for CCB
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **SAESIE**: Illegal access interrupt enable for SAES
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **PKAIE**: Illegal access interrupt enable for PKA
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **RNGIE**: Illegal access interrupt enable for RNG
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **HASHIE**: Illegal access interrupt enable for HASH
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **AESIE**: Illegal access interrupt enable for AES
0: Interrupt disabled
1: Interrupt enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADC12IE**: Illegal access interrupt enable for ADC1 and ADC2
0: Interrupt disabled
1: Interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGIE**: Illegal access interrupt enable for ICACHE registers
0: Interrupt disabled
1: Interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCIE**: Illegal access interrupt enable for TSC
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **CRCIE**: Illegal access interrupt enable for CRC
0: Interrupt disabled
1: Interrupt enabled

Bits 2:0 Reserved, must be kept at reset value.

5.3.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged access only. This register is used to enable interrupt of illegal access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZIC1IE	TZSC1IE	Res.	Res.	Res.	Res.	Res.	TAMPIE	RTCIE	EXTIIE	SYSCFGIE	RCCIE	PWRIE	FLASHIE	FLASH_REGIE	GPDMA1IE
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MPCBB2_REGIE**: Illegal access interrupt enable for MPCBB2 registers
0: Interrupt disabled
1: Interrupt enabled

Bit 26 **SRAM2IE**: Illegal access interrupt enable for SRAM2
0: Interrupt disabled
1: Interrupt enabled

Bit 25 **MPCBB1_REGIE**: Illegal access interrupt enable for MPCBB1 registers
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **SRAM1IE**: Illegal access interrupt enable for SRAM1
0: Interrupt disabled
1: Interrupt enabled

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **TZIC1IE**: Illegal access interrupt enable for GTZC1 TZIC registers
0: (B_0x0) interrupt disabled
1: (B_0x1) interrupt enabled

Bit 14 **TZSC1IE**: Illegal access interrupt enable for GTZC1 TZSC registers
0: Interrupt disabled
1: Interrupt enabled

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **TAMPIE**: Illegal access interrupt enable for TAMP
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **RTCIE**: Illegal access interrupt enable for RTC
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **EXTIIE**: Illegal access interrupt enable for EXTI
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **SYSCFGIE**: Illegal access interrupt enable for SYSCFG
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RCCIE**: Illegal access interrupt enable for RCC
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **PWRIE**: Illegal access interrupt enable for PWR
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **FLASHIE**: Illegal access interrupt enable for FLASH memory
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **FLASH_REGIE**: Illegal access interrupt enable for FLASH registers
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **GPDMA1IE**: Illegal access interrupt enable for GPDMA1
0: Interrupt disabled
1: Interrupt enabled

5.3.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFF	OPAMPF	I3C1F	SPI3F	Res.	Res.	FDCAN1F	LPTIM2F	Res.
							rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSF	I2C2F	I2C1F	UART5F	UART4F	USART3F	Res.	SPI2F	IWDGF	WWDGF	TIM7F	TIM6F	Res.	TIM4F	TIM3F	TIM2F
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VREFBUFF**: Illegal access flag for VREFBUF
0: No illegal access event
1: Illegal access event

Bit 23 **OPAMPF**: Illegal access flag for OPAMP
0: No illegal access event
1: Illegal access event

Bit 22 **I3C1F**: Illegal access flag for I3C1
0: No illegal access event
1: Illegal access event

Bit 21 **SPI3F**: Illegal access flag for SPI3
0: No illegal access event
1: Illegal access event

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **FDCAN1F**: Illegal access flag for FDCAN1 and FDCAN SRAM
0: No illegal access event
1: Illegal access event

Bit 17 **LPTIM2F**: Illegal access flag for LPTIM2
0: No illegal access event
1: Illegal access event

Bit 16 Reserved, must be kept at reset value.

Bit 15 **CRSF**: Illegal access flag for CRS
0: No illegal access event
1: Illegal access event

Bit 14 **I2C2F**: Illegal access flag for I2C2
0: No illegal access event
1: Illegal access event

Bit 13 **I2C1F**: Illegal access flag for I2C1
0: No illegal access event
1: Illegal access event

Bit 12 **UART5F**: Illegal access flag for UART5
0: No illegal access event
1: Illegal access event

Bit 11 **UART4F**: Illegal access flag for UART4
0: No illegal access event
1: Illegal access event

Bit 10 **USART3F**: Illegal access flag for USART3
0: No illegal access event
1: Illegal access event

Bit 9 Reserved, must be kept at reset value.

Bit 8 **SPI2F**: Illegal access flag for SPI2
0: No illegal access event
1: Illegal access event

Bit 7 **IWDGF**: Illegal access flag for IWDG
0: No illegal access event
1: Illegal access event

Bit 6 **WWDGF**: Illegal access flag for WWDG
0: No illegal access event
1: Illegal access event

Bit 5 **TIM7F**: Illegal access flag for TIM7
0: No illegal access event
1: Illegal access event

Bit 4 **TIM6F**: Illegal access flag for TIM6
0: No illegal access event
1: Illegal access event

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TIM4F**: Illegal access flag for TIM4
0: No illegal access event
1: Illegal access event

Bit 1 **TIM3F**: Illegal access flag for TIM3
0: No illegal access event
1: Illegal access event

Bit 0 **TIM2F**: Illegal access flag for TIM2
0: No illegal access event
1: Illegal access event

5.3.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM4F	LPTIM3F	LPTIM1F	I2C3F	LPUART1F	I3C2F	USBF	Res.	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	Res.	SPI1F	TIM1F
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **COMPF**: Illegal access flag for COMP
0: No illegal access event
1: Illegal access event

Bit 15 **LPTIM4F**: Illegal access flag for LPTIM4
0: No illegal access event
1: Illegal access event

Bit 14 **LPTIM3F**: Illegal access flag for LPTIM3
0: No illegal access event
1: Illegal access event

Bit 13 **LPTIM1F**: Illegal access flag for LPTIM1
0: No illegal access event
1: Illegal access event

Bit 12 **I2C3F**: Illegal access flag for I2C3
0: No illegal access event
1: Illegal access event

Bit 11 **LPUART1F**: Illegal access flag for LPUART1
0: No illegal access event
1: Illegal access event

Bit 10 **I3C2F**: Illegal access flag for I3C2
0: No illegal access event
1: Illegal access event

Bit 9 **USBF**: Illegal access flag for USB
0: No illegal access event
1: Illegal access event

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SAI1F**: Illegal access flag for SAI1
0: No illegal access event
1: Illegal access event

Bit 6 **TIM17F**: Illegal access flag for TIM17
0: No illegal access event
1: Illegal access event

Bit 5 **TIM16F**: Illegal access flag for TIM16
0: No illegal access event
1: Illegal access event

Bit 4 **TIM15F**: Illegal access flag for TIM15
0: No illegal access event
1: Illegal access event

Bit 3 **USART1F**: Illegal access flag for USART1
0: No illegal access event
1: Illegal access event

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SPI1F**: Illegal access flag for SPI1
0: No illegal access event
1: Illegal access event

Bit 0 **TIM1F**: Illegal access flag for TIM1
0: No illegal access event
1: Illegal access event

5.3.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1F	DAC1F	RAMCFGF	Res.	OCTOSPI1_REGF	Res.	Res.	SDMMC1F	CCBF
							rw	rw	rw		rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESF	PKAF	RNGF	HASHF	AESF	Res.	Res.	ADC12F	Res.	ICACHE_REGF	Res.	TSCF	CRCF	Res.	Res.	Res.
rw	rw	rw	rw	rw			rw		rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADF1F**: Illegal access flag for ADF1
- 0: No illegal access event
- 1: Illegal access event

Bit 23 **DAC1F**: Illegal access flag for DAC1
- 0: No illegal access event
- 1: Illegal access event

Bit 22 **RAMCFGF**: Illegal access flag for RAMCFG
- 0: No illegal access event
- 1: Illegal access event

Bit 21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1_REGF**: Illegal access flag for OCTOSPI1 registers
- 0: No illegal access event
- 1: Illegal access event

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC1F**: Illegal access flag for SDMMC1
- 0: No illegal access event
- 1: Illegal access event

Bit 16 **CCBF**: Illegal access flag for CCB
- 0: No illegal access event
- 1: Illegal access event

Bit 15 **SAESF**: Illegal access flag for SAES
- 0: No illegal access event
- 1: Illegal access event

Bit 14 **PKAF**: Illegal access flag for PKA
- 0: No illegal access event
- 1: Illegal access event

Bit 13 **RNGF**: Illegal access flag for RNG
0: No illegal access event
1: Illegal access event

Bit 12 **HASHF**: Illegal access flag for HASH
0: No illegal access event
1: Illegal access event

Bit 11 **AESF**: Illegal access flag for AES
0: No illegal access event
1: Illegal access event

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **ADC12F**: Illegal access flag for ADC1 and ADC2
0: No illegal access event
1: Illegal access event

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGF**: Illegal access flag for ICACHE registers
0: No illegal access event
1: Illegal access event

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCF**: Illegal access flag for TSC
0: No illegal access event
1: Illegal access event

Bit 3 **CRCF**: Illegal access flag for CRC
0: No illegal access event
1: Illegal access event

Bits 2:0 Reserved, must be kept at reset value.

5.3.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZIC1F	TZSC1F	Res.	Res.	Res.	Res.	Res.	TAMPF	RTCF	EXTIF	SYSCFGF	RCCF	PWRF	FLASHF	FLASH_REGF	GPDMA1F
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MPCBB2_REGF**: Illegal access flag for MPCBB2 registers
0: No illegal access event
1: Illegal access event

Bit 26 **SRAM2F**: Illegal access flag for SRAM2
0: No illegal access event
1: Illegal access event

Bit 25 **MPCBB1_REGF**: Illegal access flag for MPCBB1 registers
0: No illegal access event
1: Illegal access event

Bit 24 **SRAM1F**: Illegal access flag for SRAM1
0: No illegal access event
1: Illegal access event

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **TZIC1F**: Illegal access flag for GTZC1 TZIC registers
0: No illegal access event
1: Illegal access event

Bit 14 **TZSC1F**: Illegal access flag for GTZC1 TZSC registers
0: No illegal access event
1: Illegal access event

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **TAMPF**: Illegal access flag for TAMP
0: No illegal access event
1: Illegal access event

Bit 7 **RTCF**: Illegal access flag for RTC
0: No illegal access event
1: Illegal access event

Bit 6 **EXTIF**: Illegal access flag for EXTI
0: No illegal access event
1: Illegal access event

Bit 5 **SYSCFGF**: Illegal access flag for SYSCFG
0: No illegal access event
1: Illegal access event

Bit 4 **RCCF**: Illegal access flag for RCC
0: No illegal access event
1: Illegal access event

Bit 3 **PWRF**: Illegal access flag for PWR
0: No illegal access event
1: Illegal access event

Bit 2 **FLASHF**: Illegal access flag for FLASH memory
0: No illegal access event
1: Illegal access event

Bit 1 **FLASH_REGF**: Illegal access flag for FLASH registers
0: No illegal access event
1: Illegal access event

Bit 0 **GPDMA1F**: Illegal access flag for GPDMA1
0: No illegal access event
1: Illegal access event

5.3.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CVREFBUFF	COPAMPF	CI3C1F	CSPI3F	Res.	Res.	CFDCAN1F	CLPTIM2F	Res.
							rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRSF	CI2C2F	CI2C1F	CUART5F	CUART4F	CUSART3F	Res.	CSPI2F	CIWDGF	CWWDGF	CTIM7F	CTIM6F	Res.	CTIM4F	CTIM3F	CTIM2F
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CVREFBUFF**: Clear the illegal access flag for VREFBUF
- 0: No action
- 1: Status flag cleared

Bit 23 **COPAMPF**: Clear the illegal access flag for OPAMP
- 0: No action
- 1: Status flag cleared

Bit 22 **CI3C1F**: Clear the illegal access flag for I3C1
- 0: No action
- 1: Status flag cleared

Bit 21 **CSPI3F**: Clear the illegal access flag for SPI3
- 0: No action
- 1: Status flag cleared

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **CFDCAN1F**: Clear the illegal access flag for FDCAN1 and FDCAN SRAM.
- 0: No action
- 1: Status flag cleared

Bit 17 **CLPTIM2F**: Clear the illegal access flag for LPTIM2
- 0: No action
- 1: Status flag cleared

Bit 16 Reserved, must be kept at reset value.

Bit 15 **CCRSF**: Clear the illegal access flag for CRS
- 0: No action
- 1: Status flag cleared

Bit 14 **CI2C2F**: Clear the illegal access flag for I2C2
- 0: No action
- 1: Status flag cleared

Bit 13 **CI2C1F**: Clear the illegal access flag for I2C1
- 0: No action
- 1: Status flag cleared

Bit 12 **CUART5F**: Clear the illegal access flag for UART5
0: No action
1: Status flag cleared

Bit 11 **CUART4F**: Clear the illegal access flag for UART4
0: No action
1: Status flag cleared

Bit 10 **CUSART3F**: Clear the illegal access flag for USART3
0: No action
1: Status flag cleared

Bit 9 Reserved, must be kept at reset value.

Bit 8 **CSPI2F**: Clear the illegal access flag for SPI2
0: No action
1: Status flag cleared

Bit 7 **CIWDGF**: Clear the illegal access flag for IWDG
0: No action
1: Status flag cleared

Bit 6 **CWWDGF**: Clear the illegal access flag for WWDG
0: No action
1: Status flag cleared

Bit 5 **CTIM7F**: Clear the illegal access flag for TIM7
0: No action
1: Status flag cleared

Bit 4 **CTIM6F**: Clear the illegal access flag for TIM6
0: No action
1: Status flag cleared

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CTIM4F**: Clear the illegal access flag for TIM4
0: No action
1: Status flag cleared

Bit 1 **CTIM3F**: Clear the illegal access flag for TIM3
0: No action
1: Status flag cleared

Bit 0 **CTIM2F**: Clear the illegal access flag for TIM2
0: No action
1: Status flag cleared

5.3.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOMPF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C3F	CLPUART1F	CI3C2F	CUSBF	Res.	CSAI1F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	Res.	CSPI1F	CTIM1F
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CCOMPF**: Clear the illegal access flag for COMP
- 0: No action
- 1: Status flag cleared

Bit 15 **CLPTIM4F**: Clear the illegal access flag for LPTIM4
- 0: No action
- 1: Status flag cleared

Bit 14 **CLPTIM3F**: Clear the illegal access flag for LPTIM3
- 0: No action
- 1: Status flag cleared

Bit 13 **CLPTIM1F**: Clear the illegal access flag for LPTIM1
- 0: No action
- 1: Status flag cleared

Bit 12 **CI2C3F**: Clear the illegal access flag for I2C3
- 0: No action
- 1: Status flag cleared

Bit 11 **CLPUART1F**: Clear the illegal access flag for LPUART1
- 0: No action
- 1: Status flag cleared

Bit 10 **CI3C2F**: Clear the illegal access flag for I3C2
- 0: No action
- 1: Status flag cleared

Bit 9 **CUSBF**: Clear the illegal access flag for USB
- 0: No action
- 1: Status flag cleared

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSAI1F**: Clear the illegal access flag for SAI1
- 0: No action
- 1: Status flag cleared

Bit 6 **CTIM17F**: Clear the illegal access flag for TIM17
0: No action
1: Status flag cleared

Bit 5 **CTIM16F**: Clear the illegal access flag for TIM16
0: No action
1: Status flag cleared

Bit 4 **CTIM15F**: Clear the illegal access flag for TIM15
0: No action
1: Status flag cleared

Bit 3 **CUSART1F**: Clear the illegal access flag for USART1
0: No action
1: Status flag cleared

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CSPI1F**: Clear the illegal access flag for SPI1
0: No action
1: Status flag cleared

Bit 0 **CTIM1F**: Clear the illegal access flag for TIM1
0: No action
1: Status flag cleared

5.3.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADF1F	CDAC1F	CRAMCFGF	Res.	COCTOSPI1_REGF	Res.	Res.	CSDMMC1F	CCCBF
							rw	rw	rw		rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSAESF	CPKAF	CRNGF	CHASHF	CAESF	Res.	Res.	CADC12F	Res.	CICACHE_REGF	Res.	CTSCF	CCRCF	Res.	Res.	Res.
rw	rw	rw	rw	rw			rw		rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CADF1F**: Clear the illegal access flag for ADF1
0: No action
1: Status flag cleared

Bit 23 **CDAC1F**: Clear the illegal access flag for DAC1
0: No action
1: Status flag cleared

Bit 22 **CRAMCFGF**: Clear the illegal access flag for RAMCFG
0: No action
1: Status flag cleared

Bit 21 Reserved, must be kept at reset value.

Bit 20 **COCTOSPI1_REGF**: Clear the illegal access flag for OCTOSPI1 registers
0: No action
1: Status flag cleared

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CSDMMC1F**: Clear the illegal access flag for SDMMC1
0: No action
1: Status flag cleared

Bit 16 **CCCBF**: Clear the illegal access flag for CCB
0: No action
1: Status flag cleared

Bit 15 **CSAESF**: Clear the illegal access flag for SAES
0: No action
1: Status flag cleared

Bit 14 **CPKAF**: Clear the illegal access flag for PKA
0: No action
1: Status flag cleared

Bit 13 **CRNGF**: Clear the illegal access flag for RNG
0: No action
1: Status flag cleared

Bit 12 **CHASHF**: Clear the illegal access flag for HASH
0: No action
1: Status flag cleared

Bit 11 **CAESF**: Clear the illegal access flag for AES
0: No action
1: Status flag cleared

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **CADC12F**: Clear the illegal access flag for ADC1 and ADC2
0: No action
1: Status flag cleared

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CICACHE_REGF**: Clear the illegal access flag for ICACHE registers
0: No action
1: Status flag cleared

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CTSCF**: Clear the illegal access flag for TSC
0: No action
1: Status flag cleared

Bit 3 **CCRCF**: Clear the illegal access flag for CRC
0: No action
1: Status flag cleared

Bits 2:0 Reserved, must be kept at reset value.

5.3.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x002C

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTZIC1F	CTZSC1F	Res.	Res.	Res.	Res.	Res.	CTAMPF	CRTCF	CEXTIF	CSYSCFGF	CRCCF	CPWRF	CFLASHF	CFLASH_REGF	CGPDMA1F
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **CMPCBB2_REGF**: Clear the illegal access flag for MPCBB2 registers
0: No action
1: Status flag cleared

Bit 26 **CSRAM2F**: Clear the illegal access flag for SRAM2
0: No action
1: Status flag cleared

Bit 25 **CMPCBB1_REGF**: Clear the illegal access flag for MPCBB1 registers
0: No action
1: Status flag cleared

Bit 24 **CSRAM1F**: Clear the illegal access flag for SRAM1
0: No action
1: Status flag cleared

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **CTZIC1F**: Clear the illegal access flag for GTZC1 TZIC registers
0: No action
1: Status flag cleared

Bit 14 **CTZSC1F**: Clear the illegal access flag for GTZC1 TZSC registers
0: No action
1: Status flag cleared

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **CTAMPF**: Clear the illegal access flag for TAMP
0: No action
1: Status flag cleared

Bit 7 **CRTCF**: Clear the illegal access flag for RTC
0: No action
1: Status flag cleared

Bit 6 **CEXTIF**: Clear the illegal access flag for EXTI
0: No action
1: Status flag cleared

Bit 5 **CSYSCFGF**: Clear the illegal access flag for SYSCFG
0: No action
1: Status flag cleared

Bit 4 **CRCCF**: Clear the illegal access flag for RCC
0: No action
1: Status flag cleared

Bit 3 **CPWRF**: Clear the illegal access flag for PWR
0: No action
1: Status flag cleared

Bit 2 **CFLASHF**: Clear the illegal access flag for FLASH memory
0: No action
1: Status flag cleared

Bit 1 **CFLASH_REGF**: Clear the illegal access flag for FLASH registers
0: No action
1: Status flag cleared

Bit 0 **CGPDMA1F**: Clear the illegal access flag for GPDMA1
0: No action
1: Status flag cleared

5.3.13 GTZC1 TZIC register map

Table 27. GTZC1 TZIC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZIC_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFIE	OPAMPIE	I3C1IE	SPI3IE	Res.	Res.	FDCAN1IE	LPTIM2IE	Res.	CRSIE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	Res.	SPI2IE	IWDGIE	WWDGIE	TIM7IE	TIM6IE	Res.	TIM4IE	TIM3IE	TIM2IE
	Reset value								0	0	0	0			0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0
0x004	GTZC1_TZIC_IER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPIE	LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C3IE	LPUARTIE	I3C2IE	USBIE	Res.	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	Res.	SPI1IE	TIM1IE
	Reset value																0	0	0	0	0	0	0	0		0	0	0	0	0		0	0
0x008	GTZC1_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1IE	DAC1IE	RAMCFGIE	Res.	OCTOSPI1_REGIE	Res.	Res.	SDMMC1IE	CCBIE	SAESIE	PKAIE	RNGIE	HASHIE	AESIE	Res.	Res.	ADC12IE	Res.	ICACHE_REGIE	Res.	TSCIE	CRCIE	Res.	Res.	Res.
	Reset value								0	0	0		0			0	0	0	0	0	0	0			0		0		0	0			
0x00C	GTZC1_TZIC_IER4	Res.	Res.	Res.	Res.	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZIC1IE	TZSC1IE	Res.	Res.	Res.	Res.	Res.	TAMPIE	RTCIE	EXTI1IE	SYSCFGIE	RCCIE	PWR1IE	FLASHIE	FLASH_REGIE	GPDMA1IE
	Reset value					0	0	0	0									0	0						0	0	0	0	0	0	0	0	0

Table 27. GTZC1 TZIC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	GTZC1_TZIC_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFBUFF	OPAMPF	I3C1F	SPI3F	Res.	Res.	FDCAN1F	LPTIM2F	Res.	CRSF	I2C2F	I2C1F	UART5F	UART4F	USART3F	Res.	SPI2F	IWDGF	WWDGF	TIM7F	TIM6F	Res.	TIM4F	TIM3F	TIM2F
	Reset value								0	0	0	0			0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0
0x014	GTZC1_TZIC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPF	LPTIM4F	LPTIM3F	LPTIM1F	I2C3F	LPUARTF	I3C2F	USBF	Res.	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	Res.	SPI1F	TIM1F
	Reset value																0	0	0	0	0	0	0	0		0	0	0	0	0		0	0
0x018	GTZC1_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1F	DAC1F	RAMCFGF	Res.	OCTOSPI1_REGF	Res.	Res.	SDMMC1F	CCBF	SAESF	PKAF	RNGF	HASHF	AESF	Res.	Res.	ADC12F	Res.	ICACHE_REGF	Res.	TSCF	CRCF	Res.	Res.	Res.
	Reset value								0	0	0		0			0	0	0	0	0	0	0			0		0		0	0			
0x01C	GTZC1_TZIC_SR4	Res.	Res.	Res.	Res.	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZIC1F	TZSC1F	Res.	Res.	Res.	Res.	Res.	TAMPF	RTCF	EXTIF	SYSCFGF	RCCF	PWR1F	FLASHF	FLASH_REGF	GPDMA1F
	Reset value					0	0	0	0									0	0						0	0	0	0	0	0	0	0	0
0x020	GTZC1_TZIC_FCR 1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CVREFBUFF	COPAMPF	CI3C1F	CSPI3F	Res.	Res.	CFDCAN1F	CLPTIM2F	Res.	CCRSF	CI2C2F	CI2C1F	CUART5F	CUART4F	CUSART3F	Res.	CSPI2F	CIWDGF	CWWDGF	CTIM7F	TIM6FC	Res.	CTIM4F	CTIM3F	CTIM2F
	Reset value								0	0	0	0			0	0		0	0	0	0	0	0		0	0	0	0	0		0	0	0
0x024	GTZC1_TZIC_FCR 2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOMPF	CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C3F	CLPUARTF	CI3C2F	CUSBF	Res.	CSAI1F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	Res.	CSPI1F	CTIM1F
	Reset value																0	0	0	0	0	0	0	0		0	0	0	0	0		0	0
0x028	GTZC1_TZIC_FCR 3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADF1F	CDAC1F	CRAMCFGF	Res.	COCTOSPI1_REGF	Res.	Res.	CSDMMC1F	CCCBF	CSAESF	CPKAF	CRNGF	CHASHF	CAESF	Res.	Res.	CADC12F	Res.	CICACHE_REGF	Res.	CTSCF	CCRCF	Res.	Res.	Res.
	Reset value								0	0	0		0			0	0	0	0	0	0	0			0		0		0	0			
0x02C	GTZC1_TZIC_FCR 4	Res.	Res.	Res.	Res.	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTZIC1F	CTZSC1F	Res.	Res.	Res.	Res.	Res.	CTAMPF	CRTCF	CEXTIF	CSYSCFGF	CRCCF	CPWR1F	CFLASHF	CFLASH_REGF	CGPDMA1F
	Reset value					0	0	0	0									0	0						0	0	0	0	0	0	0	0	0

Refer to *Table 23: GTZC1 subblocks address offset*.

5.4 GTZC1 MPCBB1 registers

5.4.1 GTZC1 SRAM1 MPCBB control register (GTZC1_MPCBB1_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bit 31 **SRWILADIS**: Secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction accesses a nonsecure block of the block-based SRAM (secure fetch on nonsecure block is always considered illegal).

0: Enabled, secure read/write access not allowed on nonsecure SRAM block

1: Disabled, secure read/write access allowed on nonsecure SRAM block

Bit 30 **INVSECSTATE**: SRAM1 clock security state

This bit is used to define the internal SRAM clock control in the RCC as secure or not.

0: SRAM clocks are secure if a secure area exists in the MPCBB, and nonsecure if there is no secure area.

1: SRAM clocks are nonsecure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Lock the control register of the MPCBB until next reset

This bit is cleared by default and once set. It cannot be reset until system reset.

0: Control register not locked

1: Control register locked

5.4.2 GTZC1 SRAM1 MPCBB configuration lock register 1 (GTZC1_MPCBB1_CFGLOCKR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SPLCK 11	SPLCK 10	SPLCK 9	SPLCK 8	SPLCK 7	SPLCK 6	SPLCK 5	SPLCK 4	SPLCK 3	SPLCK 2	SPLCK 1	SPLCK 0
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **SPLCKi**: Security/privilege configuration lock for super-block i (i = 11 to 0)
This bit is set by software, and can be cleared only by system reset.
0: GTZC1_MPCBB1_SECCFGRi and GTZC1_MPCBB1_PRIVCFGRi can be written.
1: Writes to GTZC1_MPCBB1_SECCFGRi and GTZC1_MPCBB1_PRIVCFGRi are ignored.

5.4.3 GTZC1 SRAM1 MPCBB security configuration for super-block x register (GTZC1_MPCBB1_SECCFGRx)

Address offset: 0x100 + 0x4 * x (x = 0 to 11)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x00000000 when TZEN = 0. Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECi**: Security configuration for block i (i = 31 to 0)
Unprivileged write to this bit is ignored if PRIVi bit is set in GTZC1_MPCBB1_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBB1_CFGLOCKR1.
0: Nonsecure access only to block i, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block i, belonging to super-block x.

5.4.4 GTZC1 SRAM1 MPCBB privileged configuration for super-block x register (GTZC1_MPCBB1_PRIVCFGRx)

Address offset: 0x200 + 0x4 * x (x = 0 to 11)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x00000000 when TZEN = 0. Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVi**: Privileged configuration for block i, belonging to super-block x (i = 31 to 0)
Nonsecure write to this bit is ignored if SECi bit is set in GTZC1_MPCBB1_SECCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBB1_CFGLOCKR1.
0: Privileged and unprivileged access to block i, belonging to super-block x
1: Only privileged access to block i, belonging to super-block x

5.4.5 GTZC1 MPCBB1 register map

Table 28. GTZC1 MPCBB1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_MPCBB1_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC1_MPCBB1_CFGLOCKR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK11	SPLCK10	SPLCK9	SPLCK8	SPLCK7	SPLCK6	SPLCK5	SPLCK4	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0FC	Reserved	Reserved																															
0x100 + 0x4 * x (x=0 to 11)	GTZC1_MPCBB1_SECCFGRx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x200 + 0x4 * x (x=0 to 11)	GTZC1_MPCBB1_PRIVCFGRx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 5.0.3: GTZC implementation*.

5.5 GTZC1 MPCBB2 registers

All registers are accessed only by words (32-bit).

5.5.1 GTZC1 SRAM2 MPCBB control register (GTZC1_MPCBB2_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
															rs

Bit 31 **SRWILADIS**: Secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction accesses a nonsecure block of the block-based SRAM (secure fetch on nonsecure block is always considered illegal).

0: Enabled, secure read/write access not allowed on nonsecure SRAM block

1: Disabled, secure read/write access allowed on nonsecure SRAM block

Bit 30 **INVSECSTATE**: SRAM2 clock security state

This bit is used to define the internal SRAM clock control in the RCC as secure or not.

0: SRAM clocks are secure if a secure area exists in the MPCBB, and nonsecure if there is no secure area.

1: SRAM clocks are nonsecure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: Lock the control register of the MPCBB until next reset

This bit is cleared by default and once set, it cannot be reset until system reset.

0: Control register not locked

1: Control register locked

5.5.2 GTZC1 SRAM2 MPCBB configuration lock register 1 (GTZC1_MPCBB2_CFGLOCKR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK3	SPLCK2	SPLCK1	SPLCK0
												rs	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SPLCKi**: Security/privilege configuration lock for super-block i (i = 3 to 0)
This bit is set by software, and can be cleared only by system reset.
0: GTZC1_MPCBB2_SECCFGRi and GTZC1_MPCBB2_PRIVCFGRi can be written.
1: Writes to GTZC1_MPCBB2_SECCFGRi and GTZC1_MPCBB2_PRIVCFGRi are ignored.

5.5.3 GTZC1 SRAM2 MPCBB security configuration for super-block x register (GTZC1_MPCBB2_SECCFGRx)

Address offset: 0x100 + 0x4 * x (x = 0 to 3)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x00000000 when TZEN = 0. Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECi**: Security configuration for block i (i = 31 to 0)
Unprivileged write to this bit is ignored if PRIVi bit is set in GTZC1_MPCBB2_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBB2_CFGLOCKR1.
0: Nonsecure access only to block i, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block i, belonging to super-block x.

5.5.4 GTZC1 SRAM2 MPCBB privileged configuration for super-block x register (GTZC1_MPCBB2_PRIVCFGRx)

Address offset: 0x200 + 0x4 * x (x = 0 to 3)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 1. The reset value is 0x00000000 when TZEN = 0. Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVi**: Privileged configuration for block i, belonging to super-block x (i = 31 to 0)
Nonsecure write to this bit is ignored if SECi bit is set in GTZC1_MPCBB2_SECCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBB2_CFGLOCKR1.
0: Privileged and unprivileged access to block i, belonging to super-block x
1: Only privileged access to block i, belonging to super-block x

5.5.5 GTZC1 MPCBB2 register map

Table 29. GTZC1 MPCBB2 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_MPCBB2_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC1_MPCBB2_CFGLOCKR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value																													0	0	0	0
0x014-0x0FC	Reserved	Reserved																															
0x100 + 0x4 * x (x=0 to 3)	GTZC1_MPCBB2_SECCFGRx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x200 + 0x4 * x (x=0 to 3)	GTZC1_MPCBB2_PRIVCFGRx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 5.0.3: GTZC implementation*.

6 RAM configuration controller (RAMCFG)

6.1 RAMCFG introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1and SRAM2).

6.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Parity check on SRAM2
- Write protection (1-Kbyte granularity) on SRAM2
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

- The STM32U3 devices feature 256 Kbytes SRAM, made of several blocks that can be powered down in Stop mode to reduce consumption:
  - SRAM1: 2 × 16-Kbyte + 5 × 32-Kbyte blocks (total 192 Kbytes)
  - SRAM2: 8-Kbyte + 24-Kbyte + 32-Kbyte blocks (total 64 Kbytes). In addition SRAM2 blocks can be retained in Standby mode according RRSB1, RRSB2, and RRSB3 bits setting in PWR_CR1.
  - Refer to power control (PWR_CR1 and PWR_CR2) for more details.
- All internal SRAMs are erased by hardware in case of Readout protection (RDP) level regression to level 0.5 or level 0. Refer to *Section 7.6.2: Readout protection (RDP)* for more details.
- Both SRAM1 and SRAM2 are erased when a system reset occurs if the corresponding SRAMx_RST option bit is selected in the flash memory user option bytes. Refer to *Section 7.4: FLASH option bytes* for more details.
- SRAM2 is protected by the tamper detection circuit, and is erased by hardware in case of tamper detection. Refer to *Section 44: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds the registers related to the internal SRAMs parity check, write protection and software erase.
- Parity check can be enabled for SRAM2.
- Both SRAM1 and SRAM2 can be erased by software.
- Write protection can be set on SRAM2, with 1-Kbyte granularity.

The table below summarizes the features supported by each internal SRAM.

Table 30. Internal SRAM features

SRAM feature	SRAM1 (192 Kbytes)	SRAM2 (64 Kbytes)
Optional retention in Standby mode	-	X
Erased with RDP regression	X	X
Erased or blocked by tamper detection	-	X
Optionally erased with system reset	X	X
Software erase	X	X
Parity check	-	X
Write protection	-	X

6.3.2 Parity error detection (SRAM2)

The parity error is supported by SRAM2. For each word, 4 bits are used for parity check (1 bit per byte) in order to increase memory robustness, as required for instance by Class B or SIL norms.

Parity can be controlled through PCE bit (RAMCFG_M2CR) and RAMCFG_M2PARKEYR:

- To enable parity: if parity is disabled, enable it by setting PCE bit.
- To disable parity: if parity is enabled, disable it with the following sequence:
  - Write 0xAE into PARKEY[7:0] (RAMCFG_M2PARKEYR).
  - Write 0x75 into PARKEY[7:0] (RAMCFG_M2PARKEYR).
  - Write 0 into PCE bit (1 to 0 transition is possible only if PARKEY[7:0] is unlocked).

When parity is enabled by SRAM2_PE user option bit, the PCE bit is automatically set (in RAMCFG_M2CR) after system reset. Refer to *Section 7.4: FLASH option bytes* for more details.

An interrupt or NMI is generated if enabled by the PEIE or PENMI bit in RAMCFG_M2IER. The failing address is stored in the PEA[31:0] in RAMCFG_M2PEAR if the ALE bit is set in RAMCFG_M2CR.

The parity error triggers a system break event in TIM1/15/16/17, if the SPL bit is set in SYSCFG_CFGR2.

6.3.3 Write protection (SRAM2)

The SRAM2 is made of 64 1-Kbyte pages. Each 1-Kbyte page can be write-protected by setting its corresponding PxWP (x = 0 to 63) bit in the RAMCFG_M2WPRx.

6.3.4 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in RAMCFG_MxERKEYR.
2. Write 0x53 in RAMCFG_MxERKEYR.
3. Write 1 to the SRAMER bit in RAMCFG_MxCR.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is ongoing.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is written while an erase is ongoing, wait states are inserted on the AHB bus until the end of the erase operation. The SRAM is read as zero while an erase is ongoing.

6.4 RAMCFG in low-power modes

Table 31. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit the sleep mode.
Stop	The content of RAMCFG registers is kept. The parity check on SRAM2 is functional and parity error interrupt or NMI causes the device to exit from Stop 0, Stop1 and Stop 2 modes.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby.

6.5 RAMCFG interrupts

The table below gives the list of RAMCFG interrupt requests.

Table 32. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
RAMCFG	Parity error	PED	PEIE = 1 and PENMI = 0	Write 1 in CPED	Yes	Yes(1)	No
NMI	Parity error	PED	PENMI	Write 1 in CPED	Yes	Yes(1)	No

1. Stop 0, Stop 1, and Stop 2 modes only.

6.6 RAMCFG registers

In registers described below, x refers to SRAM1 or SRAM2 when x = 1 or 2 respectively.

6.6.1 RAMCFG memory 1 control register (RAMCFG_M1CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rs								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY bitfield of RAMCFG_M1ERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

6.6.2 RAMCFG memory 1 interrupt status register (RAMCFG_M1ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

6.6.3 RAMCFG memory 2 control register (RAMCFG_M2CR)

Address offset: 0x40

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	PCE
							rs				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY bitfield of RAMCFG_M2ERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable in case of parity error

0: Failing address not stored in the SRAM2 parity error address registers

1: Failing address stored in the SRAM2 parity error address registers

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PCE**: Parity check enable

This bit reset value is defined by the user option bit configuration. When set, this bit can be cleared by software only after writing the unlock sequence in RAMCFG_M2PARKEYR.

0: Parity check disabled

1: Parity check enabled

6.6.4 RAMCFG memory 2 interrupt enable register (RAMCFG_M2IER)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PENMI	Res.	Res.	PEIE
												rs			rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PENMI**: Parity error NMI
This bit is set by software and cleared only by a global RAMCFG reset.
0: NMI not generated in case of Parity error
1: NMI generated in case of Parity error
Note: if this bit is set, the RAMCFG maskable interrupt is not generated whatever PEIE value.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **PEIE**: Parity error interrupt enable
0: Parity error interrupt disabled
1: Parity error interrupt enabled

6.6.5 RAMCFG memory 2 interrupt status register (RAMCFG_M2ISR)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	PED	Res.
							r							r	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation
0: No erase operation ongoing
1: Erase operation ongoing

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **PED**: Parity error detected
0: No Parity error detected
1: Parity error detected

Bit 0 Reserved, must be kept at reset value.

6.6.6 RAMCFG memory 2 parity error address register (RAMCFG_M2PEAR)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PEA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PEA[31:0]**: Parity error address

When the ALE bit is set in RAMCFG_MxCR, this bitfield is updated with the address corresponding to the parity error.

6.6.7 RAMCFG memory 2 interrupt clear register (RAMCFG_M2ICR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPED	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CPED**: Clear parity error detected

Writing 1 to this flag clears PED bit in RAMCFG_MxISR. Reading this flag returns the PED value.

Bit 0 Reserved, must be kept at reset value.

6.6.8 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 31 to 0)

These bits are set by software and cleared only by a global RAMCFG reset.

0: Write protection of SRAM2 1-Kbyte page y is disabled.

1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.9 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.10 RAMCFG memory 2 parity key register (RAMCFG_M2PARKEYR)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARKEY[7:0]**: Parity write protection key
The following steps are required to unlock the write protection of PCE in RAMCFG_MxCR.
1) Write 0xAE into PARKEY[7:0].
2) Write 0x75 into PARKEY[7:0].
Note: Writing a wrong key reactivates the write protection.

6.6.11 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x28 + 0x40 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_MxCR.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.12 RAMCFG register map

Table 33. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RAMCFG_M1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x04	Reserved	Reserved																															
0x08	RAMCFG_M1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x0C-0x24	Reserved	Reserved																															
0x28	RAMCFG_M1ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C-0x3C	Reserved	Reserved																															
0x40	RAMCFG_M2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	PCE
	Reset value																								0				0				x
0x44	RAMCFG_M2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PENMI	Res.	Res.	PEIE
	Reset value																													0			0
0x048	RAMCFG_M2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	PED	Res.
	Reset value																								0							0	
0x4C	Reserved	Reserved																															
0x050	RAMCFG_M2PEAR	PEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	RAMCFG_M2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	Res.
	Reset value																															0	
0x058	RAMCFG_M2WPR1	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	RAMCFG_M2WPR2	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	Reserved	Reserved																															

Table 33. RAMCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x064	RAMCFG_M2PARKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x068	RAMCFG_M2ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

7 Embedded flash memory (FLASH)

7.1 FLASH introduction

The flash memory interface manages accesses to the flash memory, maximizing throughput to the CPU, instruction cache, and DMAs. It implements the flash memory erase and program operations, as well as the read and write protection mechanisms. It also implements the security and privileged access control features. It is optimized in terms of power consumption with dedicated modes when the MCU is in low-power modes.

7.2 FLASH main features

- Up to 1 Mbyte of flash memory supporting read-while-write capability (RWW)
- Memory organization
  - Dual bank architecture (bank 1 and bank 2)
  - Main memory: up to 512 Kbytes per bank
  - Information block: 104.5 Kbytes
- 64-bit wide data read with prefetch
- Standard and burst programming modes
- Read operations
- Program and erase operations only in range 1
- 10 kcycles endurance on all flash memory
- Page erase, bank erase and mass erase (both banks)
- Bank swapping: the user flash memory address mapping of each bank can be swapped.
- Product security activated by TrustZone option bit (TZEN)
- Device life cycle managed by readout protection option byte (RDP)
- 4 write protection areas (two per bank)
- TrustZone support:
  - 2 secure areas (1 per bank)
  - 2 secure HDP (hide protection) areas part of the secure areas (one per bank)
  - 2 secure HDP (hide protection) extension areas part of the secure areas (one per bank)
- Configurable protection against unprivileged accesses with flash page granularity
- Error code correction: 8-bit ECC per 64-bit double-word allowing 2-bit error detection and 1-bit error correction
- Option-byte loader
- Advanced low-power modes (low-power read mode, bank power-down mode)

7.3 FLASH functional description

7.3.1 Flash memory organization

The flash memory has the following main features:

- Capacity up to 1 Mbyte
- Dual-bank mode:
  - Up to 512 Kbytes per bank for main memory
  - 4-Kbyte page size
  - 72-bit wide data read and write (64 effective bits plus 8 ECC bits)
  - Page, bank and mass erase

The flash memory is organized as follows:

- Main memory block organized as two banks of up to 512 Kbytes each containing up to 128 pages of 4 Kbytes
- An information block containing:
  - 68 Kbytes for system memory. This area is immutable and reserved for use by STMicroelectronics. It contains the bootloader that is used to reprogram the flash memory through one of the user communication interfaces such as USB (DFU). The system memory is programmed by STMicroelectronics when the device is manufactured. For further details, refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).
  - 36 Kbytes immutable secure area containing the root security services (RSS and RSS library) developed by STMicroelectronics.
  - 512-byte OTP (one-time programmable) for user data (64 double-words). The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double-word cannot be written anymore, even with the value 0x0000 0000 0000 0000.
  - Option bytes for user configuration. Unlike user flash memory and system memory, it is not mapped to any memory address and can be accessed only through the FLASH register interface.

The memory organization is based on a main area and an information block as shown in the table below.

Table 34. Flash module 1-Mbyte dual bank-organization

Flash memory area		Flash memory address	ADDR_ECC	Size (bytes)	Name
Main memory	Bank 1	0x0800 0000 - 0x0800 0FFF	0x00000 - 0x00FFF	4K	Page 0
		0x0800 1000 - 0x0800 1FFF	0x01000 - 0x01FFF	4K	Page 1
		...	...	...	...
		0x0807 F000 - 0x0807 FFFF	0x7F000 - 0x7FFFF	4K	Page 127
	Bank 2	0x0808 0000 - 0x0808 0FFF	0x00000 - 0x00FFF	4K	Page 0
		0x0808 1000 - 0x0808 1FFF	0x01000 - 0x01FFF	4K	Page 1
		...	...	...	...
		0x080F F000 - 0x080F FFFF	0x07F000 - 0x7FFFF	4K	Page 127

Table 34. Flash module 1-Mbyte dual bank-organization (continued)

Flash memory area		Flash memory address	ADDR_ECC	Size (bytes)	Name
Nonsecure information block	Bank 1	0x0BF8 F000 - 0x0BF9 77FF	0x04800 - 0x0CFFF	34K	System memory
	Bank 2	0x0BF9 7800 - 0x0BF9 FFFF	0x04800 - 0x0CFFF	34K	
	Bank 1	0x0BFA 0000 - 0x0BFA 01FF	0x0D000 -0x0D1FF	512	OTP area
Secure information block(1)	Bank 1	0x0FF8 0000 - 0x0FF8 3BFF	0x00000 - 0x03BFF	15K	RSS
	Bank 1	0x0FF8 3C00 - 0x0FF8 47FF	0x03C00 - 0x047FF	3K	RSS library
	Bank 2	0x0FF8 4800 - 0x0FF8 8FFF	0x00000 - 0x047FF	18K	RSS library

1. The secure information block is only accessible when TrustZone is active.

7.3.2 Error code correction (ECC)

Data in flash memory are 72-bit words: 8 bits are added per double-word (64-bit).
The ECC mechanism supports:

- One error detection and correction
- Two errors detection

When one error is detected and corrected, the ECCC flag (ECC correction) is set in FLASH_ECCCORR. If the ECCCIE bit is set, an interrupt is generated.

When two errors are detected, the ECCD flag (ECC detection) is set in FLASH_ECCDETR. In this case, a NMI is generated.

When an ECCC (respectively ECCD) error is detected, the address of the failing double-word and its associated bank are saved in ADDR_ECC[18:0] and BK_ECC in FLASH_ECCCORR (respectively FLASH_ECCDETR). ADDR_ECC[2:0] are always cleared.

When ECCC (respectively ECCD) is set, ADDR_ECC[18:0] and BK_ECC are not updated if a new ECCC (respectively ECCD) error occurs. FLASH_ECCCORR (respectively FLASH_ECCDETR) is updated only when ECCC (respectively ECCD) flags are cleared.

Note: For an erased flash line, one error is detected and corrected but two errors detection is not supported. When an ECC error is reported, a new read at the failing address may not generate an ECC error if the data is still present in the current buffer, even if ECCC and ECCD are cleared. As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical bank 1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

The following addresses in the system flash memory are used to store words including ECC errors to allow run-time tests by software on ECC correction detection capability:

- 0x0BFA 0F00 (embeds a word with 1-bit error)
- 0x0BFA 0F80 (embeds a word with 2-bit error)

In case the second address is read, for instance by the debugger memory viewer, a NMI is generated.

7.3.3 Read access latency

To correctly read data from flash memory, the number of wait states (latency) must be correctly programmed in FLASH_ACR, according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE} (refer to *Dynamic voltage scaling management* in PWR section).

The table below shows the correspondence between wait states and CPU clock frequency.

Table 37. Number of wait states according to CPU clock (HCLK) frequency (LPM = 0)

Wait states (WS) (latency)	HCLK (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 CPU cycle)	≤ 32	≤ 16
1 WS (2 CPU cycles)	≤ 64	≤ 32
2 WS (3 CPU cycles)	≤ 96	≤ 48

The flash memory supports a low-power read mode when setting the LPM bit in FLASH_ACR. The table below shows the correspondence between wait states and CPU clock frequency when LPM bit is set.

Table 38. Number of wait states according to CPU clock (HCLK) frequency (LPM = 1)

Wait states (WS) (latency)	HCLK (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 CPU cycle)	≤ 20	≤ 16
1 WS (2 CPU cycles)	≤ 40	≤ 32
2 WS (3 CPU cycles)	≤ 60	≤ 48
3 WS (4 CPU cycles)	≤ 80	NA
4 WS (5 CPU cycles)	≤ 96	NA

After reset, the CPU clock frequency is 12 MHz, 1 wait state (WS) is configured in FLASH_ACR, and the normal read mode is selected (LPM = 0).

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-bus and through the instruction cache if it is enabled. The prefetch block aims at increasing the efficiency of C-bus accesses in case the instruction cache is enabled, by reducing the cache refill latency. Prefetch is efficient in case of sequential code. Prefetch in the flash memory allows the next sequential instruction line to be read from the flash memory while the current instruction line is being filled in instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in FLASH_ACR. This bit must be set only if at least 1 wait state is needed to access the flash memory.

Note: Prefetch tends to increase the code execution performance at the cost extra flash memory accesses. It must be used carefully in low-power applications.

When changing the CPU frequency, the software sequences detailed below must be applied in order to tune the number of wait states needed to access the flash memory.

Increase the CPU frequency

1. Program the new number of wait states to the LATENCY bits in FLASH_ACR.
2. Check that the new number of wait states is taken into account to access the flash memory by reading back FLASH_ACR.
3. Modify the CPU clock source by writing the SW bits in RCC_CFGR1.
4. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
5. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account, by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in RCC_CFGR1 and RCC_CFGR2.

Decrease the CPU frequency

1. Modify the CPU clock source by writing the SW bits in RCC_CFGR1.
2. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
3. Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in RCC_CFGR1 and RCC_CFGR2.
4. Program the new number of wait states to the LATENCY bits in FLASH_ACR.
5. Check that the new number of wait states is used to access the flash memory by reading back FLASH_ACR.

The software sequences detailed below must be applied in order to modify the read mode.

From normal read mode to low-power read mode

1. Set the LPM bit in FLASH_ACR.
2. Check that the low-power read mode is activated by reading FLASH_ACR.

From low-power read mode to normal read mode

1. Reset the LPM bit in FLASH_ACR.
2. Check that the normal read mode is activated by reading FLASH_ACR.

7.3.4 Bank power-down mode

After reset, both banks are in normal mode. In order to reduce power consumption, each bank can be independently put in power-down mode by setting the PDREQx bit in FLASH_ACR.

Request entry in power-down mode for bank x

1. Check that bank x is not in power-down mode and no request to put it in power-down mode is pending (PDx bit in FLASH_SR and PDREQx bit in FLASH_ACR must be reset).
2. Unlock FLASH_PDKEYxR with correct keys.
3. Set the PDREQx bit in FLASH_ACR.
4. Check that the PDx bit in FLASH_SR is set. The PDREQx bit in FLASH_ACR is automatically reset, and FLASH_PDKEYxR is locked.

Note: If bank x is currently being accessed, the power-down request is delayed until the access is completed.

Requesting power-down entry for a bank already in power-down mode has no effect. The PDREQx bit in FLASH_ACR is automatically reset and FLASH_PDKEYxR is locked.

Return to normal mode

Any access to a bank in power-down mode automatically wakes up the bank. A penalty of 5 µs minimum is taken to wake up the bank.

Wake up bank 1 (respectively bank 2) is done in one of the following cases:

- Upon a valid read access to bank 1 (resp. bank 2)
- Upon a valid write access to bank 1(resp. bank 2)
- Upon a valid bank erase on bank 1 (resp. bank 2)

Waking up both bank 1 and bank 2 is done in one of the following cases:

- Upon a valid mass erase
- Upon an option-byte modification
- Upon an option-byte loading
- Upon system reset

Note: The software can reduce the flash bank wake-up time by enabling HSI16 before waking up the bank.

7.3.5 Flash memory program and erase operations

The embedded flash memory can be programmed using in-circuit programming or in-application programming.

The in-circuit programming (ICP) method is used to update the entire contents of the flash memory, using the JTAG, SWD protocol, or the bootloader to load the user application into the MCU. The ICP offers quick and efficient design iterations, and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, the in-application programming (IAP) can use any communication interface supported by the microcontroller (such as I/Os, USB, CAN, UART, I2C,or SPI) to download programming data into the memory. The IAP can be used to reprogram the flash memory while the application is running. Nevertheless, part of the application must have been previously programmed in the flash memory using ICP.

An ongoing flash memory operation does not block the CPU as long as the CPU does not access the same flash memory bank. Code or data fetches are possible on one bank while a write/erase operation is performed to the other bank (refer to *Section 7.3.9*).

On the contrary, during a program/erase operation to the flash memory, any attempt to read the same flash memory bank stalls the bus. The read operation proceeds correctly once the program/erase operation has been completed.

The MCU supports TrustZone that defines secure and nonsecure areas in the flash memory. All program and erase operations can be performed in secure mode through secure registers, or in nonsecure mode through nonsecure registers (refer to *Section 7.5*).

Program, erase and option change operations are only allowed in range 1. An error is reported when these operations are launched in range 2.

Unlock the secure/nonsecure flash control registers

After reset, write is not allowed in flash control registers (FLASH_SCR and FLASH_CR) in order to protect the flash memory against possible unwanted operations due (for example, electric disturbances).

The following sequence is used to unlock these registers:

1. Write KEY1 = 0x4567 0123 in FLASH_SKEYR or FLASH_KEYR.
2. Write KEY2 = 0xCDEF 89AB in FLASH_SKEYR or FLASH_KEYR.

Any wrong sequence locks up FLASH_SCR or FLASH_CR until the next system reset. In the case of a wrong key sequence, a bus error is detected and a hard fault interrupt is generated.

FLASH_CR (resp. FLASH_SCR) can be locked again by software by setting the LOCK bit in FLASH_CR (resp. FLASH_SCR).

Note: FLASH_CR and FLASH_SCR cannot be written when the BSY bits are set. Any attempt to write them with the BSY bits set, causes the AHB bus to stall until the BSY bits are cleared.

Wait for data-to-write flags (WDW)

WDW flags in FLASH_SR and FLASH_SSR are both set when a secure or nonsecure write access has been done in the write buffer. They are cleared when the BSY flags are set (meaning that the write buffer is freed and the programming operation actually starts in the flash memory), or in case of error.

It is software responsibility to ensure that the two words in the same double-word are all written.

FLASH secure and nonsecure busy flags

BSY flags in FLASH_SR and FLASH_SSR are both set when a secure or nonsecure FLASH operation is started:

- Erase operation: setting the STRT bit in FLASH_CR or FLASH_SCR
- Write operation: setting the PG bit in FLASH_CR or FLASH_SCR and writing a double-word in the flash memory
- Option-byte programming: setting the OPTSTRT (in FLASH_CR)

7.3.6 Flash main memory erase sequences

The flash memory erase operation can be performed at page level, bank level or on the whole flash memory (mass erase). Mass erase does not affect the information block (system flash, OTP and option bytes). The erase operation is either secure or nonsecure.

Page erase

To erase a page, follow the procedure below:

1. Check that voltage range is set to range 1.
2. Check that no flash memory operation is ongoing by checking the BSY bit and that the write buffer is empty by checking the WDW bit in the FLASH_SR or FLASH_SSR.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.

4. Set the PER bit and select the page to erase (PNB) with the associated bank (BKER) in FLASH_CR or FLASH_SCR.
5. Set the STRT bit in FLASH_CR or FLASH_SCR. Set the STRT by privilege code if targeted page is privileged.
6. Wait for the BSY bit to be cleared in FLASH_SR or FLASH_SSR.
7. Clear PER in FLASH_CR or FLASH_SCR if there is not anymore page erase request.

Bank 1 or bank 2 mass erase

To perform a bank mass erase, follow the procedure below:

1. Check that voltage range is set to range 1.
2. Check that no flash memory operation is ongoing by checking the BSY bit. Check that the write buffer is empty by checking the WDW bit in FLASH_SR or FLASH_SSR.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
4. Set the MER1 or MER2 bit (depending on the bank) in FLASH_CR or FLASH_SCR. Both banks can be selected in the same operation. In that case, it corresponds to a mass erase.
5. Set the STRT bit in FLASH_CR or FLASH_SCR. Set the STRT by privilege code if targeted banks are privileged.
6. Wait for the BSY bit to be cleared in FLASH_SR or FLASH_SSR.
7. The MER1 or MER2 bits can be cleared if no more bank erase is requested.

Mass erase

To perform a mass erase, follow the procedure below:

1. Check that voltage range is set to range 1.
2. Check that no flash memory operation is ongoing by checking the BSY bit. Check that the write buffer is empty by checking the WDW bit in FLASH_SR or FLASH_SSR.
3. Check and clear all nonsecure error programming flags due to a previous programming. If not, the PGSERR bit is set.
4. Set the MER1 bit and MER2 bits in FLASH_CR or FLASH_SCR.
5. Set the STRT bit in the FLASH_CR or FLASH_SCR.
6. Wait for the BSY bit to be cleared in the FLASH_SR or FLASH_SSR.
7. The MER1 and MER2 bit can be cleared if no more mass erase is requested.

Note: The internal oscillator HSI16 (16 MHz) is enabled automatically when the STRT bit is set, and disabled automatically when the STRT bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR.

To erase a page, a bank or to perform a mass erase, software must have sufficient privilege (see Table 61 and Table 62).

STRT bit is cleared automatically at the end of erase or in case of error.

7.3.7 Flash memory programming sequences

The flash memory is programmed 72 bits at a time (64-bit data + 8-bit ECC).

Programming in a previously programmed address is not allowed except if the data to write is full zero, and any attempt sets the PROGERR flag in FLASH_SR or FLASH_SSR.

It is only possible to program double-word (2 × 32-bit data):

- Any attempt to write byte or half-word sets the SIZERR flag in FLASH_SR or FLASH_SSR.
- Any attempt to write a double-word that is not aligned with a double-word address sets the PGAERR flag in FLASH_SR or FLASH_SSR.

Flash programming

The flash memory programming sequence is as follows:

1. Check that voltage range is set to range 1.
2. Check that no flash main memory operation is ongoing by checking the BSY bit in FLASH_SR or FLASH_SSR.
3. Check that the write buffer is empty by checking the WDW bit in FLASH_SR or FLASH_SSR.
4. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
5. Set the PG bit in FLASH_CR or FLASH_SCR.
6. Perform the data write operation at the desired flash memory address, or in the OTP area. Only a double- word can be programmed and OTP can be only programmed in nonsecure access:
   - Write a first word at a double-word address boundary. The WDW bits in FLASH_SR and FLASH_SSR are set to indicate that more data can be written in the write buffer.
   - Write the second word in the same double-word.
7. The BSY bit gets set. WDW is reset automatically.
8. Wait until the BSY bit is cleared in FLASH_SR or FLASH_SSR. Software must make sure that BSY is set or WDW is cleared before waiting for BSY to get cleared.
9. If the EOP flag is set in FLASH_SR or FLASH_SSR (meaning that the programming operation has succeeded, and the EOPIE bit is set), it must be cleared by software.
10. Clear the PG bit in FLASH_CR or FLASH_SCR if there are no more programming requests.
11. Clear PG bit in FLASH_CR or FLASH_SCR if there is no more programming request.

Note: When the flash memory interface receives a good sequence (a double-word), programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when the PG bit is set, and disabled automatically when the PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR.

No option-byte modification nor erase request is allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected (refer to Section 7.8).

If the user needs to program only one word, the double-word must be completed with the erase value 0xFFFF FFFF to launch automatically the programming.

ECC is calculated from the double-word to program.

The OTP data cannot be erased and are writable only once. If only one bit is at 0, the entire double word is not writable anymore, even with the value 0x0.

It is only possible to write one bank/cut after the other .

Flash burst programming (16 double-words)

The flash memory burst programming sequence is as follows:

1. Check that voltage range is set to range 1.
2. Check that no flash main memory operation is ongoing by checking the BSY bit in FLASH_SR or FLASH_SSR.
3. Check that the write buffer is empty by checking the WDW bit in FLASH_SR or FLASH_SSR.
4. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
5. Set the BWR and PG bits in FLASH_CR or FLASH_SCR.
6. Perform the data write operation at the desired flash memory address, or in the OTP area. Only 16 double-words can be programmed:
   - Write a first 32-bit word in an address aligned on a 16 double-word address (multiple of 0x80). The WDW bits in FLASH_SR and FLASH_SSR are set to indicate that more data can be written in the write buffer.
   - Write the 31 other 32-bit words consecutively.
7. The BSY bit gets set. WDW is reset automatically.
8. Wait until the BSY bit is cleared in FLASH_SR or FLASH_SSR. Software must make sure that BSY is set or WDW is cleared before waiting for BSY to get cleared.
9. If the EOP flag is set in FLASH_SR or FLASH_SSR (meaning that the programming operation has succeeded and the EOPIE bit is set), it has to be cleared by software.
10. Clear the BWR and PG bits in FLASH_CR or FLASH_SCR if there is no more programming request.

Note: When the flash memory interface receives a good sequence, programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when the PG bit is set, and disabled automatically when the PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR.

No option-byte modification nor erase request is allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected (refer to Section 7.8).

7.3.8 Flash memory errors flags

Flash programming errors

Several types of errors can be detected during secure and nonsecure operations. In case of error, the flash memory operation (programming or erasing) is aborted.

The secure errors flags are only set during a secure operation and nonsecure flags are only set during a nonsecure operation.

All error flags linked to an operation error have the effect to flush the write buffer.

- **PROGERR**: secure/nonsecure programming error

  It is set when the word to program is pointing to an address:
  - Not previously erased
  - Already fully programmed to 0
  - Already partially programmed (contains 0 and 1) and the new value to program is not full zero
  - For OTP programming, when the address is already partially programmed (contains 0 and 1)
- **SIZERR**: secure/nonsecure size programming error

  Only 32-bit data can be written. SIZERR flag is set if a byte or a half-word is written.
- **PGAERR**: secure/nonsecure alignment programming error

  It is set when the first word to be programmed is not aligned with a double-word address, or the second word does not belong to the same double-word address.

  For burst programming, it is set when the first word to be programmed is not aligned on a 16 × double-word address or if the following word writes are not done at consecutive 32-bit addresses.
- **PGSERR**: programming sequence error

  It is set if one of the following conditions occurs during an erase, program, or option change programming operation:
  - A data is written when PG is cleared.
  - A program operation is requested during erase: PG is set while MER1 or MER2 or PER is set.
  - In the erase sequence, PG is set while STRT is already set.
  - In the erase sequence, if STRT is set while MER1 and MER2 and PER are cleared.
  - If page and mass erase are requested at the same time, STRT and PER are set while MER1 or MER2 is set.
  - If an operation is started while the write buffer is waiting for the next data, STRT or OPTSTRT is set while WDW is already set.
  - If STRT and OPTSTRT are set at the same time.
  - A nonsecure PGSERR is set if the non secure STRT bit is set by a secure access.
  - An erase, program, or option-byte programming (OPTSTRT) operation is launched while voltage range is set to range 2.
  - A secure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR, or PGSERR is already set due to a previous programming error.
  - A nonsecure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR, PGSERR, or OPTWERR is already set due to a previous programming error.
- **WRPERR**: write protection error
  - See *Table 55* to *Table 57* for all the conditions of WRPERR flag setting.

- **OPTWERR**: option-byte write error

  It is set if when user option bytes are modified with an invalid configuration. It is set when attempting:

  – To program an invalid secure watermark-based area (refer to *Table 42*)
  – To set or clear the TZEN option bit when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)
  – To clear the BOOT_LOCK option bit when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)
  – To modify SWAP_BANK option bit while BOOT_LOCK and TZEN are set
  – To modify ADD option bit (in FLASH_SBOOT0R) while BOOT_LOCK is set
  – To modify DUALBANK option bit while BOOT_LOCK and TZEN are set
  – To modify SECWM1Rx (resp. SECWM2Rx) in one of the following conditions:

    - while HDP1 (resp. HDP2) area access is disabled (HDP1_ACCDIS, resp. HDP2_ACCDIS, is different than 0xA3)

    - while HDP1 (resp. HDP2) extension area access is disabled (HDP1EXT_ACCDIS, resp. HDP2EXT_ACCDIS, is different than 0xA3).
  – To modify the option bytes, except the SWAP_BANK option bit, when RDP is set to two
  – To regress from RDP level 0.5 to RDP level 0
  – To modify OEM1KEYRx while RDP level is 0.5 or 1 and OEM1LOCK bit is set
  – To modify OEM2KEYRx while RDP level is 1 and OEM2LOCK bit is set
  – To regress from RDP level 1 to RDP level 0 while OEM1LOCK bit is set and a wrong OEM1KEY is shifted through JTAG or SWD
  – To regress from RDP level 1 to RDP level 0.5 while OEM2LOCK bit is set and a wrong OEM2KEY is shifted through JTAG or SWD
  – To program OEM1KEY with less than five zero.
  – To program OEM2KEY with less than five zero
  – To modify WRPxyR while its UNLOCK bit is cleared
  – To set the UNLOCK bit in the WRPxyR when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)

If an error occurs during a secure or nonsecure program or erase operation, one of the following programming error flags is set:

- Nonsecure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, OPTWRERR, or WRPERR is set in FLASH_SR.
- If the nonsecure error interrupt enable bit ERRIE is set in FLASH_CR, an interrupt is generated and the operation error flag OPERR is set in FLASH_SR.
- Secure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, or WRPERR is set in FLASH_SSR.

  If the secure error interrupt enable bit ERRIE is set in FLASH_SCR, an interrupt is generated and the operation error flag OPERR is set in FLASH_SSR.

Note: If several successive errors are detected (for example, DMA transfer to the flash memory), the error flags cannot be cleared until the end of the successive write requests.

Any programming error flushes the write buffer.

7.3.9 Read-while-write (RWW)

The flash memory is divided into two banks allowing read-while-write operations. This feature allows a read operation to be performed from one bank while erase or program operation is performed to the other bank.

Note: Write-while-write operations are not allowed. As an example, it is not possible to perform an erase operation on one bank while programming the other one.

Read from bank 1 while page erasing in bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a page erase operation on bank 2 (and vice versa).

Read from bank 1 while mass erasing bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a mass erase operation on bank 2 (and vice versa).

Read from bank 1 while programming bank 2 (or vice versa)

While executing a program code from bank 1, it is possible to perform a program operation on bank 2 (and vice versa).

Note: Due to the Cortex-M33 unified C-bus, user software must ensure to not stall C-bus with multiple consecutive writes. It is recommended to wait for the BSY flag to be cleared before programming the next double-word.

7.3.10 Power-down during flash programming or erase operation

The contents of the flash memory currently being accessed are not guaranteed if a power-down occurs during a flash memory program or erase operation.

7.3.11 Reset during flash programming or erase operation

The contents of the flash memory currently being accessed are not guaranteed if a reset occurs during a flash memory program or erase operation. The status of the flash memory can be recovered from FLASH_OPSR when a system reset occurs during a flash memory program or erase operation.

It is software responsibility to check the status of the flash memory and to take corrective actions. This must be done after each system reset before any other programming or erase operation is performed.

The table below describes the corrective action to be taken according to the status provided in the CODE_OP field of the FLASH_OPSR.

Table 39. Flash operation interrupted by a system reset

CODE_OP	Operation interrupted	Address	Bank	System flash	Corrective action
0x0	No operation	Reserved			None
0x1	Single write	ADDR_OP	BK_OP	SYSF_OP	Page erase and single write at same location
0x2	Burst Write	ADDR_OP	BK_OP	SYSF_OP	Page erase and burst write at same location
0x3	Page erase	ADDR_OP	BK_OP	Reserved	Erase same page

Table 39. Flash operation interrupted by a system reset (continued)

CODE_OP	Operation interrupted	Address	Bank	System flash	Corrective action
0x4	Bank erase	Reserved	BK_OP	Reserved	Erase same bank
0x5	Mass erase	Reserved			Mass erase
0x6	Option change	Reserved			Option change
0x7	Reserved				

Note: For single and burst write, it is mandatory to perform a page erase because the current flash memory locations may no longer be writable. Consequently, the remaining page content must be saved before page erase and restored afterwards.

For OTP write, it is not possible to perform a page erase. The OTP double-word is lost.

For burst write, ADDR_OP gives the first address of burst. The user must restart the same burst operation. For page erase, ADDR_OP gives the first address of erased page.

7.4 FLASH option bytes

7.4.1 Option-byte description

The option bytes are configured by the end user depending on the application requirements. As a configuration example, the watchdog can be selected in hardware or software mode (refer to *Section 7.4.2*). The user option bytes are accessible through the flash memory registers.

Table 40 describes the organization of all user option bytes available in the flash memory registers.

Table 40. User option-byte organization mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register FLASH_
TZEN	IO_VDDIO2_HSLV	IO_VDD_HSLV	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1_RST	NRST_SHDW	NRST_STDBY	NRST_STOP	BDRST_POR.	BOR_LEV[2:0]			RDP								OPTR
ADD[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		BOOT0R
ADD[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		BOOT1R
ADD[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	SBOOT0R

Table 40. User option-byte organization mapping (continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register FLASH_
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]							SECWM1R1
HDP1EN								Res.	HDP1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1R2
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]							WRP1AR
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]							WRP1BR
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]							SECWM2R1
HDP2EN								Res.	HDP2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2R2
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]							WRP2AR
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]							WRP2BR
OEM1KEY[31:0]																																OEM1KEYRx
OEM1KEY[63:32]																																
OEM1KEY[95:64]																																
OEM1KEY[127:96]																																
OEM2KEY[31:0]																																OEM2KEYRx
OEM2KEY[63:32]																																
OEM2KEY[95:64]																																
OEM2KEY[127:96]																																

7.4.2 Option-byte programming

Unlocking option control register

After reset, the options related to the operation bits OPTSTRT and OBL_LAUNCH in FLASH_CR, are write-protected. To run any operation on the option-byte page, the option lock bit OPTLOCK in FLASH_CR must be cleared.

The option-byte programming (OPTSTRT) can only be launched in range 1.

The following sequence is used to unlock OPTSTRT and OBL_LAUNCH bits:

1. Unlock the FLASH_CR register with the LOCK clearing sequence (refer to *Unlock the secure/nonsecure flash control registers*).
2. Write KEY1 = 0x08192A3B in FLASH_OPTKEYR.
3. Write KEY2 = 0x4C5D6E7F in FLASH_OPTKEYRR.

The user options can be protected against unwanted erase/program operations by setting the OPTLOCK bit by software.

Note: If the LOCK bit in FLASH_CR is set by software, OPTLOCK is automatically set too.

Option-byte modification sequence

To modify the user options value, follow the procedure below:

1. Check that voltage range is set to range 1.
2. Check that no flash memory operation is ongoing by checking the BSY bit. Check that the write buffer is empty by checking the WDW bit in FLASH_SR.
3. Clear OPTLOCK option lock bit with the clearing sequence described above.
4. Clear all nonsecure error flags due to a previous operation
5. Write the desired options value in the options registers.
6. Set the option start bit OPTSTRT in FLASH_CR.
7. Wait for the BSY bit to be cleared.
8. Set the OBL_LAUNCH option bit to start option-byte loading.

Note: If the OPTWERR or PGSERR error bit is set, the old option-byte values are kept.

Option-byte loading

After the BSY bit is cleared, all new options are updated into the flash memory but they are not applied to the system. They affect the system when they are loaded. Option-byte loading (OBL) is performed in two cases:

- When OBL_LAUNCH bit is set in FLASH_CR
- After a power reset (BOR reset or exit from Standby/Shutdown modes)

On system reset rising, internal option registers are copied into option registers. These registers are also used to modify the option bytes. If these registers are not modified by the user, they reflect the option state of the system.

Rules for modifying specific option bytes

Some of the option-bytes must respect specific rules before being updated with new values. These option bytes, as well as the associated constraints, are described below:

- TZEN option bit
  - TZEN can only be set on RDP level 0.
  - Deactivation of TZEN is only possible when RDP is changing from level 1 to 0.
- BOOT_LOCK option bit
  - BOOT_LOCK has only effect when TZEN is set.
  - BOOT_LOCK can be set without any constraint.
  - Deactivation of BOOT_LOCK is only possible when RDP is level 0.
- SWAP_BANK option bit
  - It cannot be modified when BOOT_LOCK and TZEN option bits are set.
- ADD (in FLASH_SBOOT0R) option bytes
  - It cannot be modified when BOOT_LOCK option bit is set.
- DUALBANK option bit
  - It cannot be modified when BOOT_LOCK and TZEN option bits are set.

- SECWMxRy option bits
  - It is not possible to modify secure (SECWMx_STRT[6:0] and SECWMx_END[6:0] option bits), and HDP (HDPx_END[6:0] and HDPxEN[7:0] option bits) area in bank x when the HDP area access is disabled (HDPx_ACCDIS ≠ 0xA3), or when the HDP extension area access is disabled (HDPxEXT_ACCDIS ≠ 0xA3).
- RDP option bits
  - Refer to *Device life cycle managed by readout protection (RDP) transitions*.
- WRPxyR option bits
  - These bits cannot be modified when their UNLOCK bit is cleared.
- UNLOCK option bits
  - These bits can be set only when regressing from RDP level 1 to level 0.

If the user option modification tries to set or modify one of the listed option bytes without following their associated rules, the option-byte modification is discarded and the OPTWERR error flag is set.

7.5 FLASH TrustZone security and privilege protections

7.5.1 TrustZone security protection

The global TrustZone system security is activated by setting the TZEN option bit in FLASH_OPTR. When TrustZone is active (TZEN = 1), the following additional security features are available:

- Secure watermark-based user options bytes defining secure and HDP areas
- Secure HDP extension areas
- Secure or nonsecure block-based areas can be configured on-the-fly after reset. This is a volatile secure area.
- An additional RDP protection: RDP level 0.5
- Erase or program operation can be performed in secure or nonsecure mode with associated configuration bit.

When the TrustZone is disabled (TZEN = 0), the above features are deactivated, and all secure registers are RAZ/WI.

Activate TrustZone security

When the TrustZone is activated (TZEN modified from 0 to 1), the secure watermark-based user option bytes are set to default secure state: all flash memory is secure, and no HDP area, as shown in the table below.

Table 41. Default secure option bytes after TZEN activation

Secure watermark option-byte values after OBL when TZEN is activated (from 0 to 1)	Security attribute
SECWMx_STRT = 0 and SECWMx_END = 0x7F	All flash memory secure
HDPxEN = 0xB4 and HDPx_END = 0	No secure HDP area

Illegal access generation

A nonsecure access to a secure flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in GTZC1_TZIC_IER2.

A nonsecure access to a secure FLASH register generates an illegal access event. An illegal access interrupt is generated if FLASH_REGIE illegal access interrupt is enabled in GTZC1_TZIC_IER2.

Deactivate TrustZone security

Deactivation of TZEN (from 1 to 0) is only possible when the RDP is changing from level 1 to level 0.

When the TrustZone is deactivated (TZEN modified from 1 to 0) after option-byte loading, the following security features are deactivated:

- Watermark-based secure area (refer to *Section 7.5.2*)
- Block-based secure area (refer to *Section 7.5.5*)
- RDP level 0.5 (refer to *Section 7.6.2*)
- Secure interrupts (refer to *Section 7.9*)
- All secure registers are RAZ/WI.

7.5.2 Watermark-based secure flash memory area protection

When TrustZone security is active (TZEN = 1), a part of the flash memory can be protected against nonsecure read and write accesses. Up to two different nonvolatile secure areas can be defined by option bytes, and can be read or written by a secure access only: one area per bank can be selected with a page granularity.

The secure areas are defined by a start-page offset and end-page offset using SECWMx_STRT and SECWMx_END (x = 1,2 for area 1 and area 2) option bytes. These offsets are defined in secure watermark registers address registers FLASH_SECWM1R1 and FLASH_SECWM1R2.

SECWMx_STRT and SECWMx_END option bytes can only be modified by secure firmware when the HDPx_ACCDIS and HDPxEXT_ACCDIS value is reseted (0xA3).
If HDPx_ACCDIS or HDPxEXT_ACCDIS is different than reset value (0xA3), SECWMx_STRT and SECWMx_END cannot be modified until next system reset.

Table 42. Secure watermark-based area

Secure watermark option-byte values (x = 1,2)	Secure watermark protection area
SECWMx_STRT > SECWMx_END	No secure area
SECWMx_STRT = SECWMx_END	One page defined by SECWMx_STRT is secure watermark-based protected
SECWMx_STRT < SECWMx_END	The area between SECWMx_STRT and SECWMx_END is secure watermark-based protected.

Caution: Switching a flash memory area from secure to nonsecure does not erase its content. User secure software must perform the needed operation to erase the secure area before

switching an area to nonsecure attribute, whenever is needed. It is also recommended to flush the instruction cache.

7.5.3 Secure hide protection (HDP)

The secure HDP area is part of the flash memory watermark-based secure area. Access to the hide protection area can be denied by setting HDPx_ACCDIS or HDPxEXT_ACCDIS (x = 1,2 for area 1 and area 2) to a value other than 0xA3 in FLASH_SECHDPCR.

When the HDPx_ACCDIS or HDPxEXT_ACCDIS bit is activated (value ≠ 0xA3), instruction fetch, data read, write and erase operations on this hide protection area are denied.
For example, software code in the secure FLASH hide protected area can be executed only once, and deny any further access to this area until next system reset. The HDPx_ACCDIS or HDPxEXT_ACCDIS bits can be reseted (HDPx_ACCDIS = 0xA3 or HDPxEXT_ACCDIS = 0xA3) only by a system reset.

When a secure HDPx area is defined (HDPxEN = 0xB4 and SECWMx_STRT ≤ HDPx_END ≤ SECWMx_END), it is forbidden to fetch outside the HDPx area until the HDPx area is closed (HDPx_ACCDIS ≠ 0xA3).

The HDP area protection are summarized in *Table 56*.

Note: It is software responsibility to take any appropriate action to protect the HDP code before deactivating the HDPxEN bits (HDPxEN = 0xB4), such as erasing the HDP area and flushing the instruction cache.

One nonvolatile secure hide protection (HDP) area per bank can be defined with a page granularity.

The secure HDP area is enabled by the HDPxEN (x = 1,2 for area 1 and area 2). When the HDPxEN bit is deactivated (HDPxEN = 0xB4), there is no HDP area. The HDPxEN bits can be modified on-the-fly by secure firmware if the HDPx_ACCDIS is set to reset value (HDPx_ACCDIS = 0xA3). If HDPx_ACCDIS is different than reset value, the HDPxEN bits and secure watermark configuration cannot be modified until next system reset.

The secure HDP area size is defined by the end-page offset using the HDPx_END option bytes, while the start- page offset is already defined by SECWMx_STRT option bytes.These offsets are defined in secure watermark registers address registers (FLASH_SECWM1R1, FLASH_SECWM1R2, FLASH_SECWM2R1, and FLASH_SECWM2R2).

For example, to protect by HDP from the address 0x0C00 4000 (included) to the address 0x0C00 5FFF (included):

- If banks are not swapped, the option-byte registers must be programmed with:
  - SECWM1_STRT = 0x4
  - HDP1_END = 0x5
- If the two banks are swapped, the protection must apply to bank 2, and the option-byte registers must be programmed with:
  - SECWM2_STRT = 0x4
  - HDP2_END = 0x5

Note: For more details on the bank swapping mechanism, refer to Section 7.5.9.

If an invalid secure HDP area is defined as described in the table below, the OPTWERR flag error is set, and option-byte modification is discarded.

Table 43. Secure hide protection

HDPx watermark option-byte values (x = 1,2)		Hide protection area
HDPxEN = 0xB4	-	No secure HDP area
HDPxEN ≠ 0xB4	SECWMx_STRT ≤ HDPx_END ≤ SECWMx_END	The area between SECWMx_STRT and HDPx_END is secure HDP protected.
	Others	Invalid secure area. Hide protection area is defined outside the secure area.

The table below summarizes the possible secure and HPD protection area configurations.

Table 44. Secure and HDP protections

Secure and HDP watermark option-byte values		Protections area
HDPxEN	**Option bytes**	
x	SECWMx_STRT > SECWMx_END	No secure area
0xB4	SECWMx_STRT ≤ SECWMx_END	No secure HDP area Secure between SECWMx_STRT and SECWMx_END If SECWMx_STRT = SECWMx_END, one page defined by SECWMx_STRT is secure protected.
≠ 0xB4	SECWMx_STRT ≤ HDPx_END ≤ SECWMx_END	The area between SECWMx_STRT and HDPx_END is secure HDP protected. If SECWMx_STRT = HDPx_END, one page defined by HDPx_END is secure HDP protected.
	Others	Invalid secure area HDP area is defined outside the secure area.

7.5.4 Secure hide protection extension (HDP extension)

The secure HDPx (x = 1,2 for area 1 and area 2) area can be extended using HDPx_EXT values in FLASH_SECHDPCR. HDPx_EXT indicates the number of pages added to the HDP area.

The HDPx extension is only composed by the secure HDPx_EXT pages which belong to the flash memory watermark-based secure area. Any pages outside the watermark-based secure area are not part of the HDPx extension area. It means the end page offset of the HDPx extension area cannot be greater than SECWMx_END.

The start page offset of the HDPx extension area depends on the presence of an HDPx area. If there is an HDPx area, the HDPx extension area starts from HDPx_END + 1. Otherwise, it starts from SECWMx_STRT.

Note: If there is no watermark-based secure area, there is no HDP extension. An HDP extension area can be defined even if there is no valid HDP area

HDP extension area access

Access to the HDPx extension area can be denied thanks to HDPxEXT_ACCDIS bits in FLASH_SECHDPCR.

- When HDPxEXT_ACCDIS value is set to the reset value (0xA3), access to the HDPx extension is granted. There is no write access restrictions to HDPx_EXT and HDPxEXT_ACCDIS bits.
- When HDPxEXT_ACCDIS value is set to 0x5C, any operations (instruction fetch, data read, write and erase) to the HDPx extension or HDPx is denied until next system reset. Moreover, HDPx extension area size can only be incremented.

  There are some write access restrictions on HDPx_EXT and HPxEXT_ACCDIS bits:
  - HDPx_EXT can only be incremented.
  - HDPxEXT_ACCDIS cannot be set to 0xA3.
- When HDPxEXT_ACCDIS value is set to a value different than 0xA3 and 0x5C, any operations (instruction fetch, data read, write and erase) to the HDPx extension or HDPx is denied until next system reset. Moreover HDPx extension area size cannot be modified until next system reset.

  There are some write access restrictions on HDPx_EXT and HPxEXT_ACCDIS bits:
  - Write access to HDPx_EXT bits are ignored.
  - Write access to HDPxEXT_ACCDIS bits are ignored

Table 45. Write access register to HDPx_EXT and HDPxEXT_ACCDIS

Current HDPxEXT_ACCDIS value	Write access to		Operation (fetch, read, write, erase) in HDPx extended area	Operation (fetch, read, write, erase) in HDPx area
	HDPxEXT_ACCDIS	HDPx_EXT		
0xA3 (reset value)	OK	OK	Allowed	Depends on HDPx_ACCDIS value
0x5C	OK if new value different than 0xA3. Otherwise, write access is ignored	OK if new value greater than current value. Otherwise, write access is ignored.	Denied	Denied
Others	Write ignored	Write ignored		

When the HDPx extension is enabled (HDPxEXT_ACCDIS ≠ 0xA3), the HDPxEN bits and secure watermark configuration cannot be modified until next system reset (refer to *Rules for modifying specific option bytes*).

One volatile secure hide protection extension (HDP extension) area per bank can be defined with a page granularity.

When a HDPx extension area is defined and it is still open (HDPxEXT_ACCDIS = 0xA3) and there is not any open HDP area, it is forbidden to fetch outside the HDPx extended area until it is closed (HDPxEXT_ACCDIS ≠ 0xA3). The HDP and HDP extension area protection are summarized in *Table 56*.

The table below summarizes the possible HDP extension area configurations.

Table 46. HDP extension protections

Secure and HDP watermark option-byte values		HDP extension area
HDPxEN	**Option bytes**	
≠ 0xB4	HDPx_END < SECWMx_END	The secure HDPx extension is composed by the following pages: HDPx_END +1 ≤ pages ≤ min (HDPx_END + HDPx_EXT , SECWMx_END)
≠ 0xB4	HDPx_END = SECWMx_END	No secure HDPx extension area
0xB4	-	The secure HDPx extension is composed by the following pages: SECWMx_STRT ≤ pages ≤ min (SECWMx_STRT - 1 + HDPx_EXT, SECWMx_END)

For example:

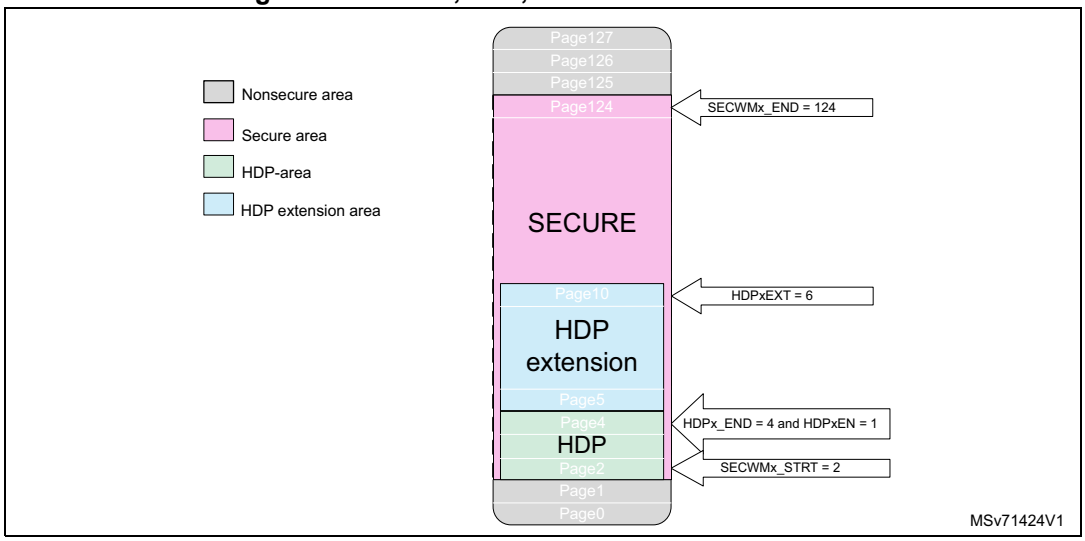
SECWM1_STRT = 0x2 SECWM1_END = 0x7C

HDP1_END = 0x4 and HDP1EN ≠ 0xB4

HDP1_EXT = 0x6

Pages from 0x2 to 0x4 (included) are protected by HDP and the pages from 0x5 to 0xA are protected by the HDP extension. This is illustrated by the figure below.

Figure 16. Secure, HDP, and HDP extension areas



Other examples:

- SECWM1_STRT = 0x6 and SECWM1_END = 0x7C

  HDP1EN = 0xB4

  HDP1_EXT = 0x6

  There is no valid HDP area. Pages from 0x6 to 0x7C are secure. The pages from 0x6 to 0xB are protected by the HDP extension.

- SECWM1_STRT = 0x6 and SECWM1_END = 0x10
  HDP1_END = 0xB and HPD1EN ≠ 0xB4
  HDP1_EXT = 0x9
  Pages from 0x6 to 0xB (included) are protected by HDP. The pages from 0xC to 0x10 are protected by the HDP extension.

7.5.5 Block-based secure flash memory area protection

Any page can be programmed on-the-fly as secure or nonsecure using the block-based configuration registers. FLASH_SECBB1Rx (resp. FLASH_SECBB2Rx) are used to configure the security attribute for pages in bank 1 (resp. bank 2).

When the page security attribute (bit i in SECBByRx) is set, the security attribute is the same as the secure watermark-based area. The secure page is only accessible by a secure access.

If the SECBByRx[i] bit is set or reset for a page already included in a secure watermark-based area, the page keeps the watermark-based protection security attributes.

To modify a block-based page security attribution, the following actions are recommended:

- Check that no flash memory operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit i in SECBByRx.

Caution: Switching a page or memory block from secure to nonsecure does not erase the content of the associated block. User secure software must perform the following needed operations before switching a block to nonsecure attribute:

- Erase page content,
- Invalidate the instruction cache.

Note: For SECBByRx bit i access control, refer to Table 63.

7.5.6 Flash security attribute state

The flash memory is secure when at least one secure area is defined either by watermark-based option bytes or block-based security registers.

It is possible to override the flash security state using the INV bit in FLASH_SCR.

FLASHEN and FLASHSMEN bits security attributes in the RCC follow the flash memory security attribute. It is possible to override the flash memory security attribute in the RCC using the INV bit in FLASH_SCR. A secure firmware setting this INV bit allows a nonsecure firmware to disable the flash memory clock when the flash memory is in power down or when the MCU enters low-power modes.

Table 47. Flash security state

Secure area	INV bit	Flash security state
None	0	Nonsecure
	1	Secure
Yes	0	Secure
	1	Nonsecure

7.5.7 Block-based privileged flash memory area protection

Any page can be programmed on-the-fly as privileged or unprivileged using the block-based configuration registers. FLASH_PRIVBB1Rx (resp. FLASH_PRIVBB2Rx) are used to configure the privilege attribute for pages in bank 1 (resp. bank 2).

When the page privilege attribute (bit i in PRIVBByRx) is set, the page is only accessible by a privileged access. An unprivileged page is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, the following actions are recommended:

- Check that no flash operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit i in PRIVBByRx.

Caution: Switching a page or memory block from privileged to unprivileged does not erase the content of the associated block.

Note: For PRIVBByRx bit i access control, refer to Table 64 and Table 65.

7.5.8 Flash memory register privileged and unprivileged modes

The flash memory registers can be read and written by privileged and unprivileged accesses, depending on the SPRIV and PRIV bits in FLASH_PRIVCFGR, with the following rules:

- When the SPRIV (resp. PRIV) bit is reset, all secure (resp. nonsecure) flash memory registers can be read and written by both privileged or unprivileged access.
- When the SPRIV (resp. PRIV) bit is set, all secure (resp. nonsecure) flash memory registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

Table 60 summarizes the flash memory registers access control.

7.5.9 Flash memory bank attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, the flash memory bank 1 is at the lower address range. When SWAP_BANK is set, the flash memory bank 1 is at the higher address range.

Flash memory bank attributes follow their bank so there is no need to modify the following registers when swapping banks:

- FLASH secure watermark y register x FLASH_SECWMyRx
- FLASH write protection x area y FLASH_WRPxyR (refer to *Section 7.6.1*)
- FLASH secure block based bank y register x FLASH_SECBByRx
- FLASH privilege block based bank y register x FLASH_PRIVBByRx
- PDREQx bits in FLASH_ACR
- PDx bits in FLASH_SR

Note: The BK_ECC bit in FLASH_ECCCORR always refers to bank 1 (resp. bank 2) when it is low (resp. high), whatever SWAP_BANK value.

The BK_OP bit in FLASH_ECCDETR always refers to bank 1 (resp. bank 2) when it is low (resp. high), whatever SWAP_BANK value.

The figures below show how security attributes and protections behave in case of bank swap.

Figure 17. Flash memory security attributes and protections in case of no bank swap (SWAP_BANK = 0)

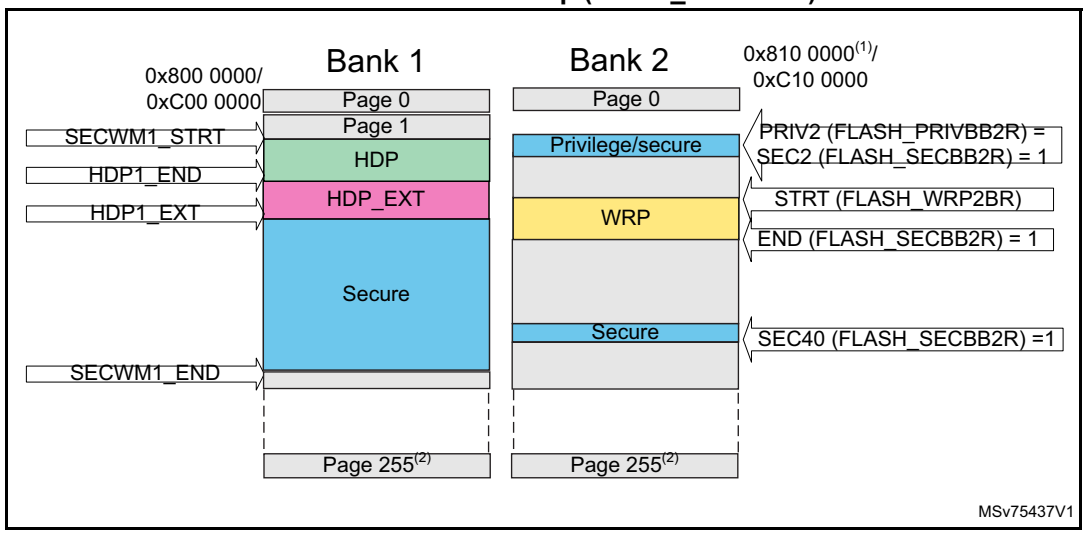
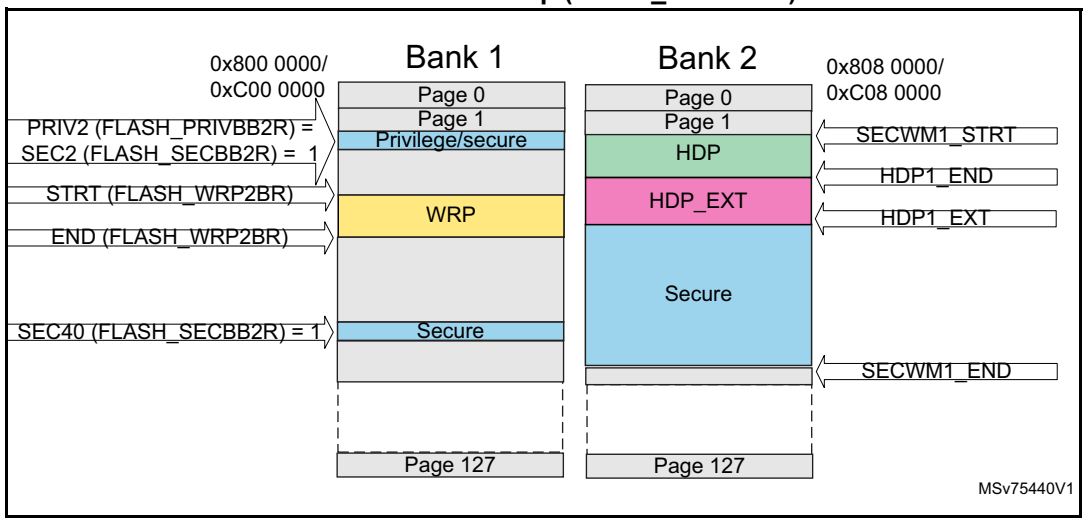


Figure 18. Flash memory security attributes and protections in case of bank swap (SWAP_BANK = 1)



7.6 FLASH memory protection

The flash memory interface implements the following protection mechanisms:

- Write protection (WRP)
- Readout protection (RDP)
- Additional secure protections when TrustZone is active (refer to *Section 7.5*)
  - Up to two secure watermark-based nonvolatile areas
  - Up to two secure hide protection areas
  - Secure block-based volatile areas with page granularity
- Privileged block-based volatile areas with page granularity (refer to *Section 7.5.7*)

7.6.1 Write protection (WRP)

The user area in the flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined in each bank, with page granularity.

Each area is defined by a start page offset and an end page offset related to the physical flash bank base address. These offsets are defined in the WRP address registers: FLASH_WRP1AR , FLASH_WRP1BR , FLASH_WRP2AR, and FLASH_WRP2BR.

The bank "x" WRP "y" area (x = 1,2 and y = A,B) is defined as follows:

- From the address: bank "x" base address + [FLASH_WRPxyR[STRT] × 0x1000] (included)
- To the address: bank "x" base address + [(FLASH_WRPxyR[END]+1) × 0x1000] (excluded)

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 3FFF (included):

- If the banks are not swapped, FLASH_WRP1AR must be programmed with:
  - STRT = 0x62
  - END = 0x73

  STRT and END in FLASH_WRP1BR can be used instead (area "B" in bank 1).
- If the two banks are swapped, the protection must apply to bank 2, and FLASH_WRP2AR must be programmed with:
  - STRT = 0x62
  - END = 0x73

  STRT and END in FLASH_WRP2BR can be used instead (area "B" in bank 2).

Note: For more details on the bank swapping mechanism, refer to Section 7.5.9.

When WRP is active, protected flash memory pages cannot be erased or programmed. Consequently, a software mass erase cannot be performed if one area is write-protected.

If an erase/program operation to a write-protected part of the flash memory is attempted, the secure or nonsecure write protection error flag (WRPERR) is set in FLASH_SR or FLASH_SSR. This flag is also set for any write access to the following:

- System flash memory
- OTP area

Note: When the memory readout protection level 1 is selected (RDP level = 1), it is not possible to program or erase the flash memory (secure or nonsecure) if the CPU debug features are

connected (JTAG or single wire) or boot code is being executed from RAM or system flash memory, even if WRP is not activated.

When the memory readout protection level 0.5 is selected (RDP level = 0.5), it is not possible to program or erase the flash secure memory if the CPU debug features are connected (JTAG or single wire), even if WRP is not activated.

To validate the WRP options, the option bytes must be reloaded through the OBL_LAUNCH bit in the flash control register.

Table 48. WRP protection

WRP registers values (x = 1/2 y = A/B)	WRP area
FLASH_WRPxy[STRT] = FLASH_WRPxy[END]	Page WRPxy is protected.
FLASH_WRPxy[STRT] > FLASH_WRPxy[END]	No WRP area
FLASH_WRPxy[STRT] < FLASH_WRPxy[END]	The pages from FLASH_WRPxy[STRT] to FLASH_WRPxy[END] are protected.

Write protection lock

Each WRP area can be independently locked by writing 0 to the UNLOCK bit in FLASH_WRP1AR, FLASH_WRP1BR, FLASH_WRP2AR, or FLASH_WRP2BR. Once a WRP area is locked, it is not possible to modify its settings. In order to unlock a WRP area, a regression to RDP level 0 must be launched.

In order to make the WRP area immutable and act as a ROM, the following actions are needed:

- If RDP level is 0, 0.5 or 1, provision a OEM1KEY in order to prevent a regression to RDP level 0 for users not knowing the key.
- If RDP level is 2, either provision a OEM1KEY (refer to first bullet) or do not provision a OEM2KEY (preventing regression from level 2 to level 1).

For more information on RDP regressions, refer to *Device life cycle managed by readout protection (RDP) transitions*.

7.6.2 Readout protection (RDP)

The readout protection protects the flash main memory, the option bytes, the backup registers, and the SRAMs. In order to reach the best protection level, it is recommended to activate TrustZone, and to set the RDP level 2 with password authentication regression enabled (refer to *Readout protection levels when TrustZone is enabled*).

Readout protection levels when TrustZone is disabled

There are three levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2).

The flash memory is protected according to the RDP option-byte value shown in the table below.

Table 49. Flash memory readout protection status (TZEN = 0)

RDP byte value	Readout protection level
0xAA	Level 0
Any value except 0xAA or 0xCC	Level 1
0xCC	Level 2

- **Level 0**: no protection

  Read, program and erase operations into the flash main memory area are possible. Option bytes, SRAMs, and backup registers are also accessible by all operations.
- **Level 1**: readout protection

  When the readout protection level 1 is set:
  - **User mode**: code executing in user mode (boot flash) can access the flash main memory, option bytes, SRAMs, and backup registers with all operations (read, erase, program).
  - **Debug, boot RAM, and bootloader modes**: in debug mode or when the microcontroller boots from RAM or system memory, the flash main memory, the backup registers, and the SRAM2 are totally inaccessible. Any read or write access to the flash main memory generates a bus error and a hard fault interrupt.
- **Level 2:** no debug

  When the readout protection level 2 is set:
  - The protection level 1 is guaranteed.
  - All debug features are disabled:

    - if OEM2 key has not been provided, JTAG and SWD are definitively disabled.

    - if OEM2 key has been provided under a lower RDP protection, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST, and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.
  - The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
  - Only boot from main flash memory is possible; all operations are allowed on the flash main memory. Read, erase and program accesses to the flash memory and SRAMs from user code are allowed.
  - Option bytes cannot be programmed nor erased except the SWAP_BANK option bit. Thus, the level 2 cannot be removed: it is an irreversible operation unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (refer to OEM2 RDP lock mechanism).

Table 50. Access status versus protection level and execution mode when TZEN = 0

Area	RDP level	User execution (boot from flash)			Debug/boot from RAM/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No(2)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory(3)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(4)	1	Yes	Yes(2)	N/A	Yes	Yes	N/A
	2	Yes	No(5)	N/A	N/A	N/A	N/A
OTP	1	Yes	Yes(6)	N/A	Yes	Yes(6)	N/A
	2	Yes	Yes(6)	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	1	Yes	Yes	N/A	No	No	N/A(8)
	2	Yes	Yes	N/A	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
4. Option bytes are only accessible through the flash registers interface and OPTSTRT bit.
5. SWAP_BANK option bit can be modified.
6. OTP can only be written once.
7. The backup registers are erased when RDP changes from level 1 to level 0.
8. All SRAMs are erased when RDP changes from level 1 to level 0. .

Readout protection levels when TrustZone is enabled

There are four levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2). The flash memory is protected according to the RDP option-byte value shown in the table below.

Table 51. Flash memory readout protection status (TZEN = 1)

RDP byte value	Readout protection level
0xAA	Level 0
0x55	Level 0.5
Any value except 0xAA or 0x55 or 0xCC	Level 1
0xCC	Level 2

- **Level 0**: no protection

  Read, program and erase operations into the flash main memory area are possible. The option bytes, the SRAMs and the backup registers are also accessible by all operations.

  – RSS mode: when booting from RSS, the debug access is disabled while executing RSS code.

- **Level 0.5**: nonsecure debug only

  All read and write operations (if no write protection is set) from/to the nonsecure flash memory are possible. The debug access to secure area is prohibited. Debug access to nonsecure area remains possible.

  – **User mode**: code executing in user mode (boot flash) can access the flash main memory, option bytes, SRAMs and backup registers with all operations (read, erase, program).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. The secure flash memory, secure backup registers, and SRAMs are inaccessible. The nonsecure flash memory, nonsecure backup registers, and nonsecure SRAMs remain accessible for debug purpose.

  – **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.

  – **Boot RAM mode**: boot from SRAM is not possible.

- **Level 1**: readout protection

  When the readout protection level 1 is set:

  – **User mode**: code executing in user mode (boot flash) can access the flash main memory, option bytes, SRAMs and backup registers with all operations (read, erase, program).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. However, an intrusion is detected in case of debug access: the flash main memory, backup registers, and the SRAM2 are totally inaccessible. Any read or write access to the flash main memory generates a bus error and a hard fault interrupt.

  – **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.

  – **Boot RAM mod**e: boot from SRAM is not possible.

- **Level 2**: no debug

  When the readout protection level 2 is set:

  – The protection level 1 is guaranteed.

  – All debug features are disabled

  -if OEM2 key has not been provided, JTAG and SWD are definitively disabled.

  – -if OEM2 key has been provided under a lower RDP protection, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST, and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.

  – The boot from SRAM (boot RAM mode) and the boot from system memory (boot loader mode) are no longer available.

  – Boot from RSS is possible.

– When booting from main flash or RSS, all operations are allowed on the flash main memory. Read, erase and program accesses to flash memory and SRAMs from user code are allowed.
– Option bytes cannot be programmed nor erased except the SWAP_BANK option bit. Thus, the level 2 cannot be removed: it is an irreversible operation unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

*STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (*refer to *OEM2 RDP lock mechanism*).

Table 52. Access status versus protection level and execution modes when TZEN = 1

Area	RDP level	User execution (boot from flash)			Debug/bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	0.5	Yes	Yes	Yes	Yes(2)	Yes(2)	Yes(2)
	1	Yes	Yes	Yes	No	No	No(5)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory(3)	0.5	Yes	No	No	Yes	No	No
	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(4)	0.5	Yes	Yes(5)	N/A	Yes	Yes(5)	N/A
	1	Yes	Yes(5)	N/A	Yes	Yes(5)	N/A
	2	Yes	No(6)	N/A	N/A	N/A	N/A
OTP	0.5	Yes	Yes(7)	N/A	Yes	Yes(7)	N/A
	1	Yes	Yes(7)	N/A	Yes	Yes(7)	N/A
	2	Yes	Yes(7)	N/A	N/A	N/A	N/A
Backup registers	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(8)
	1	Yes	Yes	N/A	No	No	N/A(8)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(9)
	1	Yes	Yes	N/A	No	No	N/A(9)
	2	Yes	Yes	N/A	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port and the bootloader mode are disabled.
2. Depends on TrustZone security access rights.
3. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
4. Option bytes are only accessible through the flash registers interface and OPTSTRT bit.
5. The flash main memory is erased when the RDP option byte regresses from level 1 to level 0.
6. SWAP_BANK option bit can be modified.
7. OTP can only be written once.

8. The backup registers are erased when RDP changes from level 1 to level 0 and when RDP changes from level 1 to level 0.5.
9. All SRAMs are erased when RDP changes from level 1 to level 0 and when RDP changes from level 1 to level 0.5.

Device life cycle managed by readout protection (RDP) transitions

It is easy to move from level 0 or level 0.5 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 0.5 or from level 1. Once in level 2, it is no longer possible to modify the readout protection level unless an OEM2 key is provisioned (refer to *OEM2 RDP lock mechanism*).

When the RDP is reprogrammed to the value 0xAA to move from level 1 to level 0, a mass erase of the flash main memory and all SRAMs is performed. Backup registers, ICACHE, and PKA SRAM are also erased. The OTP area is not erased.

At RDP level 0.5, it is not possible to request RDP level 0. Instead, a RDP increase to level 1 followed by a RDP regression to level 0 is required.

When the RDP is programmed to the value 0x55 to move from level 1 to level 0.5, a partial mass erase of the flash main memory is performed. Only nonsecure watermark-based areas are erased (even if it is defined as secure by block-based). Backup registers, ICACHE, PKA SRAM, and all SRAMs are mass erased. The OTP area is not erased. The RDP level 0.5 and partial nonsecure erase are only available when TrustZone is active.

Note: Full mass erase is performed only when level 1 is active and level 0 requested. When the protection level is increased (0 to 0.5, 0 to 1, 0.5 to 1, 1 to 2, 0 to 2 or 0.5 to 2), there is no mass erase.

To validate the readout protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in FLASH_CR.

Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to be cleared.

Figure 19. RDP level transition scheme when TrustZone is disabled (TZEN = 0)

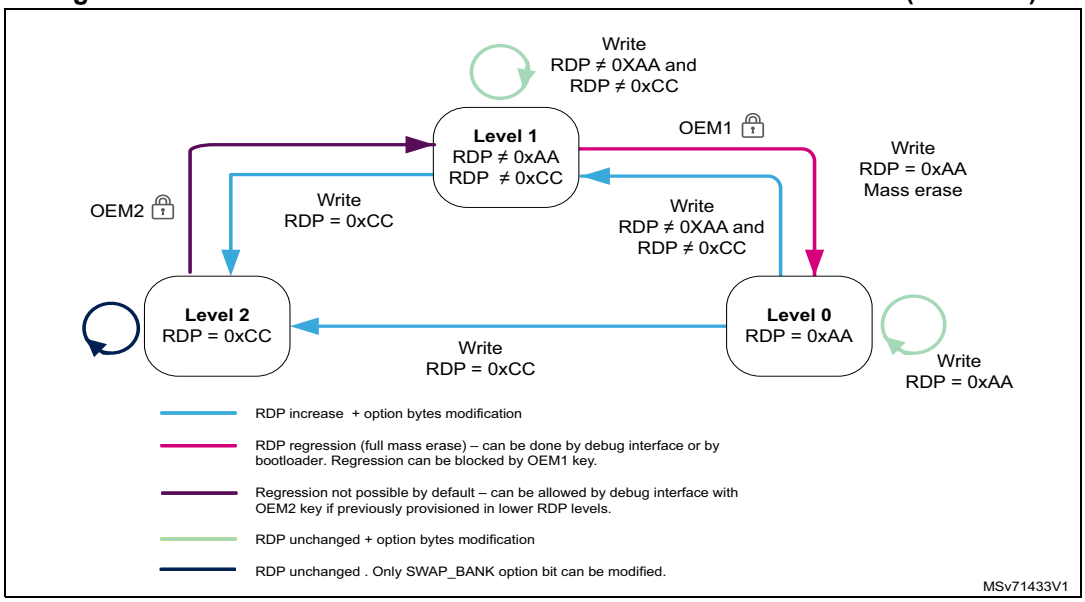
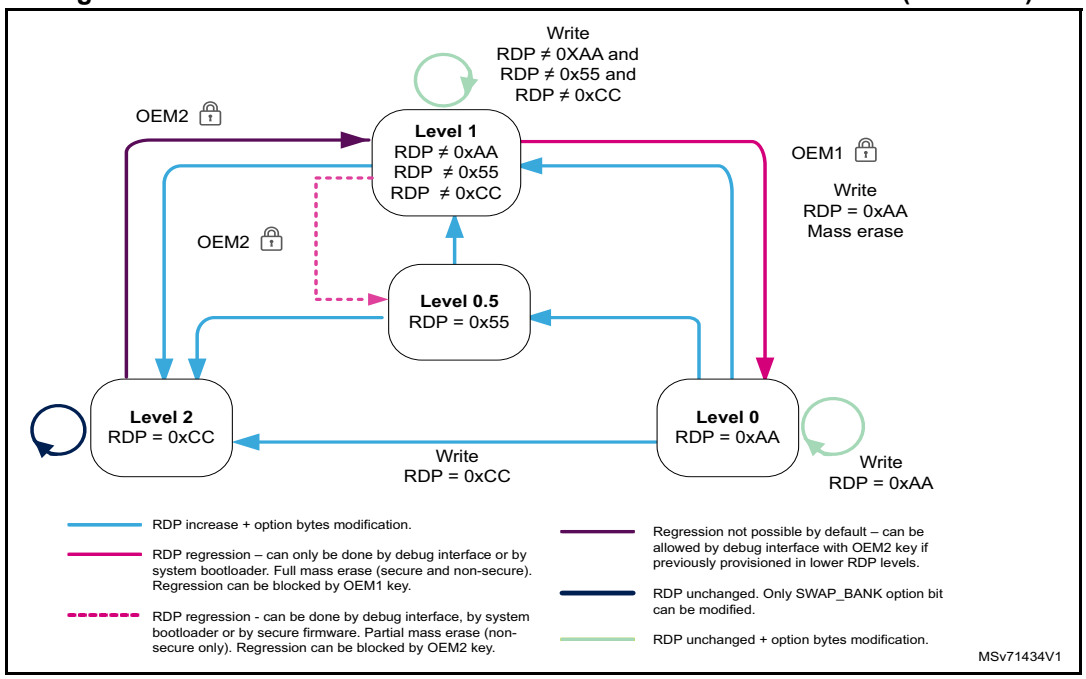


Figure 20. RDP level transition scheme when TrustZone is enabled (TZEN = 1)



OEM1/OEM2 lock activation

Two 128-bit keys (OEM1KEY and OEM2KEY) can be defined in order to lock the RDP regression. Each 128-bit key is coded on four registers:

- OEM1KEY coded on FLASH_OEM1KEYRx (x = 1 to 4)
- OEM2KEY coded on FLASH_OEM2KEYRx (x = 1 to 4).

OEM1KEY and OEM2KEY cannot be read through these registers but there are two 8-bit CRC (OEM1KEYCRC and OEM2KEYCRC), which can be used as a public key information.

Any read access to OEM1KEY and OEM2KEY returns zero.

OEM1KEY can be modified:

- In readout protection level 0
- In readout protection level 0.5 or 1 if OEM1LOCK = 0 in FLASH_SR

OEM2KEY can be modified:

- In readout protection level 0 or 0.5
- In readout protection level 1 if OEM2LOCK = 0 in FLASH_SR

When attempting to modify FLASH_OEM1KEYRx or FLASH_OEM2KEYRx without following these rules, the user option modification is not done and the OPTWERR bit is set.

In order to activate OEM1 lock mechanism, the following steps are needed:

1. Check that the OEM1LOCK bit is not set or that the readout protection is at level 0.
2. Write a 128-bit key in FLASH_OEM1KEYRx
3. Check key CRC thanks to OEM1KEYCRC.
4. Launch option modification by setting the OPTSTRT bit in FLASH_CR.
5. Wait for the BSY bit to be cleared and check that OPTWERR is not set.

6. Set the OBL_LAUNCH option bit to start option-byte loading or perform a power-on reset.
7. Check that OEM1LOCK is set.

In order to activate OEM2 lock mechanism, the following steps are needed:

1. Check that the OEM2LOCK bit is not set or that the readout protection is at level 0 or 0.5.
2. Write a 128-bit key in FLASH_OEM2KEYRx.
3. Check key CRC using OEM2KEYCRC.
4. Launch option modification by setting the OPTSTRT bit in FLASH_CR.
5. Wait for the BSY bit to be cleared and check that OPTWERR is not set.
6. Set the OBL_LAUNCH option bit to start option-byte loading or perform a power-on reset.
7. Check that OEM2LOCK is set.

Note: The OEM1KEY and OEM2KEY must contain more than four 0.

It is recommended to always activate OEM2 lock mechanism even if no regression level is expected.

Once set, the RDP transition with OEM keys mechanism cannot be disabled. This means that the OEM keys can be modified, but never cleared.

OEM1 RDP lock mechanism

The OEM1 RDP lock mechanism is active when the OEM1LOCK bit is set. It blocks the RDP level 1 to RDP level 0 regression.

In order to regress from RDP level 1 to RDP level 0, the following unlock sequence must be applied:

1. Shift OEM1KEY[31:0], OEM1KEY[63:32], OEM1KEY[95:64] and then OEM1KEY[127:96] through JTAG or SWD in DBGMCU_DBG_AUTH_HOST.
2. If this key matches the OEM1KEY value, the RDP regression can be launched by setting the OPTSTRT bit.
3. If the key does not match the OEM1KEY value, the RDP regression and any access to the flash memory are blocked until a next power-on reset.

Attempting to regress from RDP level 1 to RDP level 0 without following this sequence sets the OPTWERR option bit and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM1LOCK = 0), the regression from RDP level 1 to RDP level 0 is always granted.

OEM2 RDP lock mechanism

The OEM2 RDP lock mechanism is active when the OEM2LOCK bit is set. It allows the following actions:

- Block RDP level 1 to RDP level 0.5 regression.
- Authorize RDP level 2 to RDP level 1 regression.

In order to regress from RDP level 1 to RDP level 0.5, the following unlock sequence must be applied:

1. Shift OEM2KEY[31:0], OEM2KEY[63:32], OEM2KEY[95:64], and then OEM2KEY[127:96] through JTAG or SWD under reset in DBGMCU_DBG_AUTH_HOST.
2. Release reset if this key matches the OEM2KEY value, the RDP regression can be launched by setting the OPTSTRT bit.
3. If the key does not match the OEM2KEY value, the RDP regression and any access to the flash memory are blocked until a next power-on reset to reload option bytes.

In order to regress from RDP level 2 to RDP level 1, the following unlock sequence must be applied:

1. Shift OEM2KEY[31:0], OEM2KEY[63:32], OEM2KEY[95:64], and then OEM2KEY[127:96] through JTAG or SWD under reset in DBGMCU_DBG_AUTH_HOST register
2. Release reset if this key matches the OEM2KEY value:
   – The RDP regression is launched by hardware (it is not possible to execute instructions when the key is matching).
   – Apply a power-on reset (cycle V_{DD} power supply OFF and ON).
3. If the key does not match the OEM2KEY value, the RDP regression and any access to the flash memory are blocked until a next power-on reset to reload option bytes.

Attempting to regress from RDP level 2 to RDP level 1 without following these sequences, leaves option bytes unchanged.

Attempting to regress from RDP level 1 to RDP level 0.5 without following these sequences, sets the OPTWERR option bit and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM2LOCK = 0), the following happens:

- The regression from RDP level 1 to RDP level 0.5 is always granted.
- The regression from RDP level 2 to RDP level 1 is never granted. When attempting to modify the options bytes, the protection error flag OPTWERR is set in FLASH_SR, and an interrupt can be generated.

OEM Key CRC source

Below is the CRC source code that can be used to compare with the result of the computation provided by the flash interface in OEM1KEYCRC (resp. OEM2KEYCRC) bitfield after loading the keys in FLASH OEM1 (resp. OEM2) Key registers.

```
uint8_t getCRC(uint32_t * keyin)
{
const uint8_t CRC7_POLY = 0x7;
const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0}; uint8_t i, j,
k, crc = 0x0;
uint32_t keyval;
for (j = 0; j < 4; j++)
{
keyval = *(keyin+j);
if (j == 0) { keyval ^= key_strobe[0]; }
else { keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc; } for (i = 0, crc = 0; i < 32; i++)
```

```
{
k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1; crc <<= 1;
if (k) { crc ^= CRC7_POLY; }
}
crc^=0x55;
}
return crc;
}
```

7.7 System memory

System memory stores RSS (root secure services) firmware programmed by STMicroelectronics during production. RSS provides runtime services to user firmware. When TrustZone® is enabled (user sets the TZEN bitfield to 1), RSS provides secure services to the secure user firmware only. When TrustZone® is disabled (user sets the TZEN bitfield to 0), RSS does not provide nonsecure services to the user firmware.

7.7.1 RSS user functions

The RSS provides runtime services using the RSS library, whose functions are exposed to the user within the CMSIS device header file, provided by the STM32Cube firmware package.

RSS provides services through the RSSLIB library, dedicated to secure user firmware when the system is configured with TrustZone® enabled (TZEN bitfield set to 1).

Table 53. Macros for RSS services

TrustZone	C defined macro	Location in flash memory
Enabled	RSSLIB_PFUNC	0x0BF99040
Disabled	NA	NA

RSSLIB

The secure user firmware calls the RSSLIB functions using RSSLIB_PFUNC C-defined macro, which points to a location within the nonsecure system memory. Before calling the RSSLIB functions, the secure user firmware must define a nonsecure region above this location within SAU of the Cortex®-M33 processor, starting from RSSLIB_SYS_FLASH_NS_PFUNC_START (0x0BF99040), up to RSSLIB_SYS_FLASH_NS_PFUNC_END (0x0BF990FF). These last addresses are provided within the CMSIS device header file.

The user can configure this nonsecure region either by using the CMSIS system partition header file, or by implementing their own code for SAU setup. The CMSIS system partition header file is part of the STM32Cube firmware package.

Table 54. RSS lib interface functions

Library	Function	Attributes
RSSLIB_PFUNC	CloseExitHDP	Secure callable function
	CloseExitHDPExt	

CloseExitHDP

Secure attribute:

- secure callable function

Prototype:

- uint32_t CloseExitHDP (uint32_t HdpArea, uint32_t VectorTableAddr);

Arguments:

- HdpArea: notifies which hdp area to close, can be a combination of hdpa area 1 and hdp area 2
- VectorTableAddr: a pointer on the vector table containing the reset handler the function jumps to

Returned values:

- 0xF5F5F5F5U: error on input parameter
- otherwise, does not return

CloseExitHDP closes the requested hdp area passed in input parameter and jumps to the reset handler present within the vector table. The function does not return on successful execution.

CloseExitHDPExt

Secure attribute:

- secure callable function

Prototype:

- uint32_t CloseExitHDPExt (uint32_t HdpExtArea, uint32_t VectorTableAddr, uint32_t CloseBound);

Arguments:

- HdpExtArea: notifies which hdp extension area to close, can be a combination of hdp extension area 1 and hdp extension area 2
- VectorTableAddr: pointer on the vector table containing the reset handler the function jumps to
- CloseBound: notifies if the HDP extension area should be closed with HDPx_EXT increment allowed or not

Returned values:

- 0xF5F5F5F5U: error on input parameter
- Otherwise, does not return

CloseExitHDPExt closes the requested hdp extension area passed in input parameter and jumps to the reset handler present within the vector table. The function does not return on successful execution.

7.8 Summary of flash memory and flash memory registers access control

The flash interface evaluates the access restrictions in the following orders: RDP level, TrustZone security, write protection (for write access), HDP, and privilege. The tables below summarize all the flash memory and registers accesses status versus RDP level, WRP and HDP protections.

Table 55. Flash memory access versus RDP level when TrustZone is active (TZEN = 1)

<table>
<tr><th colspan="2" rowspan="2">Access type</th><th colspan="2">RDP level 0, RDP level 0.5, RDP level 1 no intrusion(1), or RDP level 2</th><th>RDP level 1 with intrusion(2)</th></tr>
<tr><th>Nonsecure page</th><th>Secure page</th><th>Nonsecure or secure page</th></tr>
<tr><td rowspan="4">Secure</td><td>Fetch</td><td>Bus error</td><td rowspan="2">OK</td><td rowspan="2">Bus error</td></tr>
<tr><td>Read</td><td>RAZ, flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">WI, secure WRPERR flag set, flash illegal access event</td><td rowspan="2">No WRP: OK
WRP pages: WI and secure WRPERR flag set</td><td rowspan="2">WI, secure WRPERR flag set</td></tr>
<tr><td>Page erase</td></tr>
<tr><td rowspan="4">Nonsecure</td><td>Fetch</td><td rowspan="2">OK</td><td>Bus error</td><td rowspan="2">Bus error</td></tr>
<tr><td>Read</td><td>RAZ, flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">No WRP: OK
WRP pages: WI and nonsecure WRPERR flag set</td><td rowspan="2">WI, non secure WRPERR flag set, flash illegal access event</td></tr>
<tr><td>Page erase</td><td>WI, nonsecure WRPERR flag set</td></tr>
</table>

1. RDP level 1 no intrusion = when booting from user flash memory and no debug access.
2. RDP level 1 with intrusion = when debug access detected.

Table 56. Flash memory access versus HDPx and HDPx extended (x = 1 or 2)

Access type	Flash memory area			
	HDPx (with x = 1,2)	HDP1Ext	HDP2Ext	Others
fetch	OK if HDPx is open(1) else RAZ	OK if HDP1Ext is open(2) and HDP1 and HDP2 closed or not defined, else RAZ	OK if HDP2Ext is open and HDP1 and HDP2 closed or not defined, else RAZ	OK if HDP1, HDP2, HDP1Ext and HDP2Ext are closed or not defined, else RAZ
Read	OK if HDPx is open, else RAZ	OK if HDP1Ext is open, else RAZ	OK if HDP2Ext is open, else RAZ	OK
Write/ page erase	OK if HDPx open, else WI, secure WRPERR flag set	OK if HDP1Ext is open, else WI, secure WRPERR flag set	OK if HDP2Ext is open, else WI, secure WRPERR flag set	OK

1. “HDPx area is open" means that HDPx_ACCDIS = HDPxExt_ACCDIS = 0xA3. Otherwise, it is closed. "HDPx area is defined" is described in *Table 44*.
2. HDPxExt is open" means that HDPxExt_ACCDIS = 0xA3. Otherwise, it is closed. When HDPxExt is closed (HDPxExt_ACCDIS ≠ 0xA3), HDPx is automatically closed. "HDPxExt area is not defined" is described in *Table 46*.

Table 57. Flash memory access versus RDP level when TrustZone is disabled (TZEN = 0)

Access type	RDP level 0, RDP level 1 no intrusion, or RDP level 2[1]	RDP level 1 with intrusion[2]
Fetch	OK	Bus error
Read		
Write	No WRP: OK WRP pages: WI and nonsecure WRPERR flag set	
Erase		WI and nonsecure WRPERR flag set

1. RDP level 1 no intrusion = when booting from user flash memory and no debug access.
2. RDP level 1 with intrusion = when booting from RAM or system memory or debug access detected.

Table 58. Flash memory mass erase versus RDP level when TrustZone is active (TZEN = 1)

Access type		RDP level 0, RDP level 0.5, RDP level 1 no intrusion or RDP level 2[1]					RDP level 1 with intrusion[2]
		Nonsecure flash memory	Secure flash memory			Mix nonsecure and secure flash memory	Nonsecure or secure flash memory
			HDP area (HDPxEN≠0xB4 HDPx_ ACCDIS≠0xA3)	HDP extension area (HDPxEXT_ ACCDIS≠0xA3)	Others[3]		
Secure	Bank or mass erase	WI, secure WRPERR flag set, flash memory illegal access event	WI, secure WRPERR flag set	WI, secure WRPERR flag set	No WRP: OK WRP pages: WI and secure WRPERR flag set	WI, secure WRPERR flag set, flash memory illegal access event	WI, secure WRPERR flag set
Non secure		No WRP: OK WRP pages: WI and nonsecure WRPERR flag set	WI, nonsecure WRPERR flag set, flash memory illegal access event				WI, nonsecure WRPERR flag set

1. RDP level 1 no intrusion = when booting from user flash memory and no debug access.
2. RDP level 1 with intrusion = when debug access detected.
3. “Others” refers to the other flash memory secure configurations than the one described for HDP protections or for HDP extension protection. Example: flash memory secure and HDPx area enabled but HDPx_ACCDIS = 0xA3, or flash memory secure and HDPx extension area defined but HDPxEXT_ACCDIS = 0xA3.

Table 59. Flash system memory, OTP and RSS accesses (valid for all RDP levels)

Access type		System memory (bootloader)	OTP	RSS
Secure (TZEN = 1)	Fetch	Bus error		RAZ
	Read	RAZ, flash memory register illegal access event	OK	
	Write	WI, secure WRPERR flag set, flash memory illegal access event		WI, secure WRPERR flag set

Table 59. Flash system memory, OTP and RSS accesses (valid for all RDP levels) (continued)

Access type		System memory (bootloader)	OTP	RSS
Nonsecure (TZEN = 0 or TZEN = 1)	Fetch	OK	Bus error	Bus error
	Read		OK	RAZ(1)
	Write	WI and nonsecure WRPERR flag set	OK if not virgin: WI, nonsecure PROGERR flag set	WI, nonsecure WRPERR flag set

1. Flash memory illegal access event is generated when TZEN = 1.

Table 60. FLASH registers access(1)

Access type			Nonsecure register		Secure register	
			PRIV = 1	PRIV = 0	SPRIV = 1	SPRIV = 0
Fetch	Secure/ nonsecure	Privileged/ unprivileged	Bus error			
Read/ Write	Secure(2)	Privileged	OK			
		Unprivileged	RAZ/WI	OK	RAZ/WI	OK
	Nonsecure(3)	Privileged	OK		RAZ/WI and a flash memory register illegal access event(4)	
		Unprivileged	RAZ/WI	OK		

1. Except FLASH_SECBByRx, FLASH_PRIVBByRx ,and FLASH_PRIVCFGR.
2. Secure access is only valid when TrustZone is active (TZEN = 1).
3. Nonsecure access are valid when TrustZone is active or disabled.
4. Flash register illegal access event is only generated when TZEN = 1.

Table 61. FLASH page access versus privilege mode(1)

Access type		Unprivileged page	Privileged page
Fetch, read/write, page erase	Privileged	OK	
Fetch, read	Unprivileged	OK	RAZ
Write, page erase	Unprivileged		WI, secure or nonsecure WRPERR flag set

1. When TZEN = 1, access must be granted by security firewall before privilege is considered

Table 62. FLASH mass erase versus privilege mode(1)

Access type		Unprivileged flash memory	Privileged flash memory	Mix unprivileged and privileged flash memory
Mass erase	Privileged	OK		
Mass erase	Unprivileged	OK	WI, secure or nonsecure WRPERR flag set	

1. When TZEN = 1, access must be granted by security firewall before privilege is considered

Table 63. FLASH_SECBByRx registers access when TrustZone is active (TZEN = 1)

Access type			Bit i in FLASH_PRIVBByRx	Bit i in FLASH_SECBByRx
Fetch	Secure/nonsecure	Privileged/unprivileged	-	Bus error
Read	Secure/nonsecure	Privileged/unprivileged	-	OK
Write	Secure	Privileged	-	OK
		Unprivileged	0	OK for bit i
		Unprivileged	1	WI for bit i
	Nonsecure	Privileged/unprivileged	-	WI and a flash memory register illegal access event

Table 64. FLASH_PRIVBByRx registers access when TrustZone is active (TZEN = 1)

Access type			Page secure state (watermark or blocked based)	Bit i in FLASH_PRIVBByRx
Fetch	Privileged/unprivileged	Secure/nonsecure	-	Bus error
Read	Privileged/unprivileged	Secure/nonsecure	-	OK for all bits
Write	Privileged	Secured	-	OK for all bits
		Nonsecure	Nonsecure	OK for bit i
		Nonsecure	Secure	WI for bit i
	Unprivileged	Secured/nonsecure	-	WI for all bits

Table 65. FLASH_PRIVBByRx registers access when TrustZone is disabled (TZEN = 0)

Access type		FLASH_PRIVBByRx
Fetch	Privileged/unprivileged	Bus error
Read	Privileged/unprivileged	OK
Write	Privileged	OK
	Unprivileged	WI

7.9 FLASH interrupts

Table 66. Flash interrupt requests

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH_S	Secure end of operation	Secure EOP(1)	Write secure EOP = 1	Secure EOPIE	Yes	No
	Secure operation error	Secure OPERR(2)	Write secure OPERR = 1	Secure ERRIE	Yes	No

Table 66. Flash interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH	Nonsecure end of operation	Nonsecure EOP(1)	Write nonsecure EOP = 1	Nonsecure EOPIE	Yes	No
	Nonsecure operation error	Nonsecure OPERR(2)	Write nonsecure OPERR = 1	Nonsecure ERRIE	Yes	No
	ECC correction	ECCC	Write ECCC = 1	ECCCIE	Yes	No

1. Secure EOP (resp. nonsecure EOP) is set only if secure EOPIE (resp. nonsecure EOPIE) is set.
2. Secure OPERR (resp. nonsecure OPERR) is set only if secure ERRIE (resp. nonsecure ERRIE) is set.

7.10 FLASH registers

7.10.1 FLASH access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0001

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no flash memory read is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SLEEP_PD	PDREQ2	PDREQ1	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY[3:0]			
	rw	rs	rs	rw			rw					rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **SLEEP_PD**: Flash memory power-down mode during Sleep mode

This bit determines whether the flash memory is in power-down mode or Idle mode when the device is in Sleep mode.

0: Flash in Idle mode during Sleep mode

1: Flash in power-down mode during Sleep mode

Caution: The flash must not be put in power-down while a program or an erase operation is ongoing.

Bit 13 **PDREQ2**: Bank 2 power-down mode request

This bit is write-protected with FLASH_PDKEY2R. This bit requests bank 2 to enter power-down mode. When bank 2 enters power-down mode, this bit is cleared by hardware and FLASH_PDKEY2R is locked.

0: No request for bank 2 to enter power-down mode

1: Bank 2 requested to enter power-down mode

Bit 12 **PDREQ1**: Bank 1 power-down mode request
This bit is write-protected with FLASH_PDKEY1R. This bit requests bank 1 to enter power-down mode. When bank 1 enters power-down mode, this bit is cleared by hardware and FLASH_PDKEY1R is locked.
0: No request for bank 1 to enter power-down mode
1: Bank 1 requested to enter power-down mode

Bit 11 **LPM**: Low-power read mode
This bit puts the flash memory in low-power read mode.
0: FLASH not in low-power read mode
1: FLASH in low-power read mode

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable
This bit enables the prefetch buffer in the embedded flash memory.
0: Prefetch disabled
1: Prefetch enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LATENCY[3:0]**: Latency
This bitfield represents the ratio between the HCLK (AHB clock) period and the flash memory access time.
0000: 0 wait state
0001: 1 wait state
0010: 2 wait states
...
1111: 15 wait states

7.10.2 FLASH nonsecure key register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV =1 in FLASH_PRIVCFGR register.

Access: one wait state; word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Flash memory nonsecure key
The following values must be written consecutively to unlock the FLASH_CR register, allowing flash memory nonsecure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.10.3 FLASH secure key register (FLASH_SKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

Access: one wait state; word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Flash memory secure key

The following values must be written consecutively to unlock the FLASH_SCR register, allowing flash memory secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.10.4 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x10

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: one wait state; word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Option-byte key

The following values must be written consecutively to unlock the FLASH_OPTR register allowing option-byte programming/erasing operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

7.10.5 FLASH bank 1 power-down key register (FLASH_PDKEY1R)

Address offset: 0x18

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY1[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY1[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY1[31:0]**: Bank 1 power-down key

The following values must be written consecutively to unlock PDREQ1 bit in FLASH_ACR:
KEY1_1: 0x0415 2637
KEY1_2: 0xFAFB FCFD

7.10.6 FLASH bank 2 power-down key register (FLASH_PDKEY2R)

Address offset: 0x1C

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY2[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY2[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY2[31:0]**: Bank 2 power-down key

The following values must be written consecutively to unlock PDREQ2 bit in FLASH_ACR:
KEY2_1: 0x4051 6273
KEY2_2: 0xAFBF CFDF

7.10.7 FLASH nonsecure status register (FLASH_SR)

Address offset: 0x20

Reset value: 0x000X 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD2	PD1	OEM2LOCK	OEM1LOCK	WDW	BSY
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
		rc_w1						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PD2**: Bank 2 in power-down mode

This bit indicates that the flash memory bank 2 is in power-down state. It is reset when bank 2 is in normal mode or being awaken.

Bit 20 **PD1**: Bank 1 in power-down mode

This bit indicates that the flash memory bank 1 is in power-down state. It is reset when bank 1 is in normal mode or being awaken.

Bit 19 **OEM2LOCK**: OEM2 lock

This bit indicates that the OEM2 RDP key read during the OBL is not virgin. When set, the OEM2 RDP lock mechanism is active.

Bit 18 **OEM1LOCK**: OEM1 lock

This bit indicates that the OEM1 RDP key read during the OBL is not virgin. When set, the OEM1 RDP lock mechanism is active.

Bit 17 **WDW**: Nonsecure wait data to write

This bit indicates that the flash memory write buffer has been written by a secure or nonsecure operation. It is set when the first data is stored in the buffer, and cleared when the write is performed in the flash memory.

Bit 16 **BSY**: Nonsecure busy

This bit indicates that a flash memory secure or nonsecure operation is in progress. It is set at the beginning of a flash operation, and reset when the operation finishes or when an error occurs.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OPTWERR**: Option write error

This bit is set by hardware when the options bytes are written with an invalid configuration. It is cleared by writing 1. Refer to *Section 7.3.8: Flash memory errors flags* for full conditions of error flag setting.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: Nonsecure programming sequence error

This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1. Refer to *Section 7.3.8: Flash memory errors flags* for full conditions of error flag setting.

Bit 6 **SIZERR**: Nonsecure size error

This bit is set by hardware when the size of the access is a byte or half-word during a nonsecure program sequence. Only double-word programming is allowed by means of successive word accesses. This bit is cleared by writing 1.

Bit 5 **PGAERR**: Nonsecure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a double-word address, or the second does not belong to the same double-word address. This bit is cleared by writing 1.

Bit 4 **WRPERR**: Nonsecure write protection error

This bit is set by hardware when a nonsecure address to be erased/programmed belongs to a write-protected part (by WRP, HDP, or RDP level 1) of the flash memory. This bit is cleared by writing 1. Refer to *Section 7.3.8: Flash memory errors flags* for full conditions of error flag setting.

Bit 3 **PROGERR**: Nonsecure programming error

This bit is set by hardware when a nonsecure double-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Nonsecure operation error

This bit is set by hardware when a flash memory nonsecure operation (program/erase) completes unsuccessfully. This bit is set only if nonsecure error interrupts are enabled (ERRIE = 1 in FLASH_CR). This bit is cleared by writing 1.

Bit 0 **EOP**: Nonsecure end of operation

This bit is set by hardware when one or more flash memory nonsecure operation (program/erase) has been completed successfully. This bit is set only if the nonsecure end of operation interrupts are enabled (EOPIE = 1 in FLASH_CR). This bit is cleared by writing 1.

7.10.8 FLASH secure status register (FLASH_SSR)

Address offset: 0x24

Reset value: 0x0000 0000

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDW	BSY
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **WDW**: Secure wait data to write

This bit indicates that the flash memory write buffer has been written by a secure or nonsecure operation. It is set when the first data is stored in the buffer, and cleared when the write is performed in the flash memory.

Bit 16 **BSY**: Secure busy

This bit indicates that a flash memory secure or nonsecure operation is in progress. This is set on the beginning of a flash operation, and reset when the operation finishes or when an error occurs.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: Secure programming sequence error

This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1. Refer to *Section 7.3.8: Flash memory errors flags* for full conditions of error flag setting.

Bit 6 **SIZERR**: Secure size error

This bit is set by hardware when the size of the access is a byte or half-word during a secure program sequence. Only double-word programming is allowed by means of successive word accesses.This bit is cleared by writing 1.

Bit 5 **PGAERR**: Secure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a double-word address, or the second word does not belong to the same double-word address.This bit is cleared by writing 1.

Bit 4 **WRPERR**: Secure write protection error

This bit is set by hardware when a secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP, or RDP level 1) of the flash memory.This bit is cleared by writing 1. Refer to *Section 7.3.8: Flash memory errors flags* for full conditions of error flag setting.

Bit 3 **PROGERR**: Secure programming error

This bit is set by hardware when a secure double-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Secure operation error

This bit is set by hardware when a flash memory secure operation (program/erase) completes unsuccessfully. This bit is set only if secure error interrupts are enabled (ERRIE = 1 in FLASH_SCR). This bit is cleared by writing 1.

Bit 0 **EOP**: Secure end of operation

This bit is set by hardware when one or more flash memory secure operation (program/erase) has been completed successfully. This bit is set only if the secure end of operation interrupts are enabled (EOPIE = 1 in FLASH_SCR). This bit is cleared by writing 1.

7.10.9 FLASH nonsecure control register (FLASH_CR)

Address offset: 0x28

Reset value: 0xC000 0000

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is stalled until the BSY bits are reset.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no flash memory operation is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT
rs	rs			rc_w1		rw	rw							rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Nonsecure lock

This bit is set only. When set, the FLASH_CR register is locked. It is cleared by hardware after detecting the unlock sequence in FLASH_KEYR register.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 **OPTLOCK:** Option lock

This bit is set only. When set, all bits concerning user options in FLASH_CR register are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit in FLASH_CR must be cleared before doing the unlock sequence for OPTLOCK bit.

In case of an unsuccessful unlock operation, this bit remains set until the next reset.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBL_LAUNCH**: Option-byte loading forced

When set to 1, this bit forces the option-byte reloading. This bit is cleared only when the option-byte loading is complete. It cannot be written if OPTLOCK is set.

0: Option-byte loading complete

1: Option-byte loading requested

Bit 26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Nonsecure error interrupt enable

This bit enables the interrupt generation when the OPERR bit in FLASH_SR is set to 1.

0: Nonsecure OPERR error interrupt disabled

1: Nonsecure OPERR error interrupt enabled

Bit 24 **EOPIE**: Nonsecure end of operation interrupt enable

This bit enables the interrupt generation when the EOP bit in FLASH_SR is set to 1.

0: Nonsecure EOP Interrupt disabled

1: Nonsecure EOP Interrupt enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **OPTSTRT**: Option modification start
This bit triggers an option operation when set. It can not be written if OPTLOCK bit is set. This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR.

Bit 16 **STRT**: Nonsecure start
This bit triggers a nonsecure erase operation when set. If MER1, MER2, and PER bits are reset, and the STRT bit is set (this configuration is forbidden), the PGSERR bit in FLASH_SR is set .
This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_SR.

Bit 15 **MER2**: Nonsecure bank 2 mass erase
This bit triggers the bank 2 nonsecure mass erase (all bank 2 user pages) when set.

Bit 14 **BWR**: Nonsecure burst write programming mode
When set, this bit selects the burst write programming mode.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKER**: Nonsecure bank selection for page erase
0: Bank 1 selected for nonsecure page erase
1: Bank 2 selected for nonsecure page erase

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Nonsecure page number selection
These bits select the page to erase.
0000000: page 0
0000001: page 1
...
1111111: page 127

Bit 2 **MER1**: Nonsecure bank 1 mass erase
This bit triggers the bank 1 nonsecure mass erase (all bank 1 user pages) when set.

Bit 1 **PER**: Nonsecure page erase
0: Nonsecure page erase disabled
1: Nonsecure page erase enabled

Bit 0 **PG**: Nonsecure programming
0: Nonsecure FLASH programming disabled
1: Nonsecure FLASH programming enabled

7.10.10 FLASH secure control register (FLASH_SCR)

Address offset: 0x2C

Reset value: 0x8000 0000

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access stalls until the BSY bits are reset.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no flash memory operation is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT
rs		rw				rw	rw								rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Secure lock

This bit is set only. When set, the FLASH_SCR register is locked. This bit is cleared by hardware after detecting the unlock sequence in FLASH_SKEYR register.
In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

Bit 30 Reserved, must be kept at reset value.

Bit 29 **INV:** Flash memory security state invert

This bit inverts the flash memory security state.

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Secure error interrupt enable

This bit enables the interrupt generation when the OPERR bit in FLASH_SSR is set to 1.
0: Secure OPERR error interrupt disabled
1: Secure OPERR error interrupt enabled

Bit 24 **EOPIE**: Secure end of operation interrupt enable

This bit enables the interrupt generation when the EOP bit in FLASH_SSR is set to 1.
0: Secure EOP Interrupt disabled
1: Secure EOP Interrupt enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Secure start

This bit triggers a secure erase operation when set. If MER1, MER2, and PER bits are reset, and the STRT bit is set, the PGSERR in FLASH_SSR is set (this condition is forbidden).
This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_SSR.

Bit 15 **MER2**: Secure bank 2 mass erase

This bit triggers the bank 2 secure mass erase (all bank 2 user pages) when set.

Bit 14 **BWR**: Secure burst write programming mode

When set, this bit selects the burst write programming mode.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKER**: Secure bank selection for page erase
0: Bank 1 selected for secure page erase
1: Bank 2 selected for secure page erase

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Secure page number selection
These bits select the page to erase:
0000000: page 0
0000001: page 1
...
1111111: page 127

Bit 2 **MER1**: Secure bank 1 mass erase
This bit triggers the bank 1 secure mass erase (all bank 1 user pages) when set.

Bit 1 **PER**: Secure page erase
0: Secure page erase disabled
1: Secure page erase enabled

Bit 0 **PG**: Secure programming
0: Secure FLASH programming disabled
1: Secure FLASH programming enabled

7.10.11 FLASH ECC register (FLASH_ECCCORR)

Address offset: 0x30

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ECCC	Res.	Res.	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:16]		
	rc_w1						rw		r	r			r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ECCC**: ECC correction
This bit is set by hardware when one ECC error has been detected and corrected (only if ECCC was previously cleared). An interrupt is generated if ECCIE is set. This bit is cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ECCIE**: ECC correction interrupt enable
This bit enables the interrupt generation when the ECCC bit in FLASH_ECCR is set.
0: ECCC interrupt disabled
1: ECCC interrupt enabled.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

Bit 21 **BK_ECC**: ECC fail bank

This bit indicates which bank is concerned by the ECC error correction.
0: Bank 1
1: Bank 2

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC[18:0]**: ECC fail address

This bitfield indicates which address is concerned by the ECC error correction. The address is given by bank from address 0x0 0000 to 0x7 FFF8.

7.10.12 FLASH ECC detection register (FLASH_ECCDETR)

Address offset: 0x34

Reset value: 0x0000 0000

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:16]		
rc_w1									r	r			r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD**: ECC detection

This bit is set by hardware when two ECC errors have been detected (only if ECCD was previously cleared). When this bit is set, a NMI is generated. This bit is cleared by writing 1.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **SYSF_ECC**: System flash memory ECC fail

This bit indicates that the double ECC error detection is located in the system flash memory.

Bit 21 **BK_ECC**: ECC fail bank

This bit indicates which bank is concerned by the double ECC error detection.
0: Bank 1
1: Bank 2

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_ECC[18:0]**: ECC fail address

This bitfield indicates which address is concerned by the double ECC error detection. The address is given by bank from address 0x0 0000 to 0x7 FFF8.

7.10.13 FLASH operation status register (FLASH_OPSR)

Address offset: 0x38

Reset value: 0xX0XX XXXX

System reset value: 0xX0XX XXXX

Power-on reset value: 0xX0XX XXXX

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CODE_OP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SYSF_OP	BK_OP	Res.	Res.	ADDR_OP[18:16]		
r	r	r							r	r			r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_OP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code

This bitfield indicates which flash memory operation has been interrupted by a system reset:
000: No flash operation interrupted by previous reset
001: Single write operation interrupted
010: Burst write operation interrupted
011: Page erase operation interrupted
100: Bank erase operation interrupted
101: Mass erase operation interrupted
110: Option change operation interrupted
111: Reserved

Bits 28:23 Reserved, must be kept at reset value.

Bit 22 **SYSF_OP**: Operation in system flash memory interrupted

This bit indicates that the reset occurred during an operation in the system flash memory.

Bit 21 **BK_OP**: Interrupted operation bank

This bit indicates which flash memory bank was accessed when reset occurred
0: Bank 1
1: Bank 2

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:0 **ADDR_OP[18:0]**: Interrupted operation address

This bitfield indicates which address in the flash memory was accessed when reset occurred. The address is given by bank from address 0x0 0000 to 0x7 FFF8.

7.10.14 FLASH option register (FLASH_OPTR)

Address offset: 0x40

Reset value: 0xXXXX XXXX

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0x1FEF F0AA

(register bits 0 to 31 are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN	IO_VDDIO2_HSLV	IO_VDD_HSLV	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	BL_IO_VDDIO2_HSLV[1:0]		DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
rw	rw	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRAM1_RST	NRST_SHDW	NRST_STDBY	NRST_STOP	BDRST_POR	BOR_LEV[2:0]			RDP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TZEN:** Global TrustZone security enable
0: Global TrustZone security disabled
1: Global TrustZone security enabled

Bit 30 **IO_VDDIO2_HSLV:** High-speed I/O at low V_{DDIO2} voltage configuration bit
This bit can be set only with V_{DDIO2} below 2.5 V.
0: High-speed I/O at low V_{DDIO2} voltage feature disabled (V_{DDIO2} can exceed 2.5 V)
1: High-speed I/O at low V_{DDIO2} voltage feature enabled (V_{DDIO2} remains below 2.5 V)

Bit 29 **IO_VDD_HSLV:** High-speed I/O at low V_{DD} voltage configuration bit
This bit can be set only with V_{DD} below 2.5V
0: High-speed I/O at low V_{DD} voltage feature disabled (V_{DD} can exceed 2.5 V)
1: High-speed I/O at low V_{DD} voltage feature enabled (V_{DD} remains below 2.5 V)

Bit 28 Reserved, must be kept at reset value.

Bit 27 **NBOOT0:** NBOOT0 option bit
0: NBOOT0 = 0
1: NBOOT0 = 1

Bit 26 **NSWBOOT0:** Software BOOT0
0: BOOT0 taken from the option bit NBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2_RST:** SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs

Bit 24 **SRAM2_PE:** SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled

Bits 23:22 **BL_IO_VDDIO2_HSLV[1:0]:** Enable HSLV on PG2, PG3, PG4, and PG5 when bootloader is running.
00: High-speed I/O at low V_{DDIO2} voltage feature enabled (V_{DDIO2} remains below 2.5 V)
11: High-speed I/O at low V_{DDIO2} voltage feature disabled (V_{DDIO2} can exceed 2.5 V)
Others: Reserved.

Bit 21 **DUALBANK:** Dual-bank configuration
0: Single-bank FLASH with contiguous address in bank 1
1: Dual-bank FLASH with contiguous addresses

Bit 20 **SWAP_BANK:** Swap banks
0: Bank 1 and bank 2 addresses not swapped
1: Bank 1 and bank 2 addresses swapped

Bit 19 **WWDG_SW:** Window watchdog selection
0: Hardware WWDG selected
1: Software WWDG selected

Bit 18 **IWDG_STDBY:** Independent watchdog counter freeze in Standby mode
0: IWDG counter frozen in Standby mode
1: IWDG counter running in Standby mode

Bit 17 **IWDG_STOP:** Independent watchdog counter freeze in Stop mode
0: IWDG counter frozen in Stop mode
1: IIWDG counter running in Stop mode

Bit 16 **IWDG_SW:** Independent watchdog selection
0: Hardware IWDG selected
1: Software IWDG selected

Bit 15 **SRAM1_RST:** SRAM1 erase upon system reset
0: SRAM1 erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bit 14 **NRST_SHDW:** Reset generation in Shutdown mode
0: Reset generated when entering the Shutdown mode
1: No reset generated when entering the Shutdown mode

Bit 13 **NRST_STDBY:** Reset generation in Standby mode
0: Reset generated when entering the Standby mode
1: No reset generate when entering the Standby mode

Bit 12 **NRST_STOP:** Reset generation in Stop mode
0: Reset generated when entering the Stop mode
1: No reset generated when entering the Stop mode

Bit 11 **BDRST_POR:** Backup domain reset with power-on reset
0: The backup domain is not reset after a V_{DD} domain brownout reset (BOR).
This configuration must be used when V_{BAT} supply source is different from V_{DD}.
1: The backup domain is reset after a V_{DD} domain brownout reset (BOR). This configuration must be used when the V_{BAT} and V_{DD} are connected together. It is forbidden when V_{BAT} supply source is different from V_{DD}.

Bits 10:8 **BOR_LEV[2:0]:** BOR reset level

This bitfield contains the V_{DD} supply level threshold that activates/releases the reset.
000: BOR level 0 (reset level threshold around 1.7 V)
001: BOR level 1 (reset level threshold around 2.0 V)
010: BOR level 2 (reset level threshold around 2.2 V)
011: BOR level 3 (reset level threshold around 2.5 V)
100: BOR level 4 (reset level threshold around 2.8 V)

Bits 7:0 **RDP[7:0]:** Readout protection level

0xAA: Level 0 (readout protection not active)
0x55: Level 0.5 (readout protection not active, only nonsecure debug access is possible).
Only available when TrustZone is active (TZEN = 1)
0xCC: Level 2 (chip readout protection active)
Others: Level 1 (memories readout protection active)

Note: Refer to Section 7.6.2: Readout protection (RDP) for more details.

7.10.15 FLASH nonsecure boot address 0 register (FLASH_BOOT0R)

Address offset: 0x44

Reset value: 0xXXXX XXXF

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0x0800 007F
(option bytes are loaded with values from the flash memory at reset release)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **ADD[24:0]**: Nonsecure boot base address 0

The nonsecure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes. These bits correspond to address [31:7]. ADD option bytes are selected following the BOOT0 pin or NSWBOOT0 state.
Examples:
ADD[24:0] = 0x0100000: Boot from nonsecure flash memory (0x0800 0000)
ADD[24:0] = 0x017F1E0: Boot from system memory bootloader (0x0BF8 F000)
ADD[24:0] = 0x0400000: Boot from nonsecure SRAM1 on S-Bus (0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.10.16 FLASH nonsecure boot address 1 register (FLASH_BOOT1R)

Address offset: 0x48

Reset value: 0xXXXX XXXF

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0x0BF8 F07F

(option bytes are loaded with values from the flash memory at reset release)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

Bits 31:7 **ADD[24:0]**: nonsecure boot address 1

The nonsecure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes. These bits correspond to address [31:7]. ADD option bytes are selected following the BOOT0 pin or nSWBOOT0 state.

Examples:

ADD[24:0] = 0x0100000: Boot from nonsecure flash memory (0x0800 0000)

ADD[24:0] = 0x017F1E0: Boot from system memory bootloader (0x0BF8 F000)

ADD[24:0] = 0x0400000: Boot from nonsecure SRAM1 on S-Bus (0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.10.17 FLASH secure boot address 0 register (FLASH_SBOOT0R)

Address offset: 0x4C

Reset value: 0xXXXX XXXX

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0x0C00 007C

(option bytes are loaded with values from the flash memory at reset release)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[24:9]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw							rw

Bits 31:7 **ADD[24:0]**: Secure boot base address 0

The secure boot memory address can be programmed to any address in the valid address range with a granularity of 128 bytes. This bits correspond to address [31:7] ADD option bytes are selected following the BOOT0 pin or NSWBOOT0 state.
Examples:
ADD[24:0] = 0x018 0000: Boot from secure flash memory (0x0C00 0000)
ADD[24:0] = 0x01F F000: Boot from RSS (0x0FF8 0000)
ADD[24:0] = 0x060 0000: Boot from secure SRAM1 on S-Bus (0x3000 0000)

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **BOOT_LOCK**: Boot lock

When set, the boot is always forced to base address value programmed in ADD[24:0] option bytes, whatever the boot selection option. When set, this bit can only be cleared when RDP is at level 0.

7.10.18 FLASH secure watermark1 register 1 (FLASH_SECWM1R1)

Address offset: 0x50

Reset value: 0xFFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFFFF FF80

(register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1.)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **SECWM1_END[6:0]:** End page of first secure area
This bitfield contains the last page of the secure area in bank 1.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **SECWM1_STRT[6:0]:** Start page of first secure area
This bitfield contains the first page of the secure area in bank 1.

7.10.19 FLASH secure watermark1 register 2 (FLASH_SECWM1R2)

Address offset: 0x54

Reset value: 0xXXXX FFFF

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xB480 FFFF

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDP1EN[7:0]								Res.	HDP1_END[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 **HDP1EN[7:0]**: Hide protection first area enable
0xB4: No HDP area 1
others: HDP first area enabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **HDP1_END[6:0]**: End page of first hide protection area
This bitfield contains the last page of the HDP area in bank 1.

Bits 15:0 Reserved, must be kept at reset value.

7.10.20 FLASH WRP1 area A address register (FLASH_WRP1AR)

Address offset: 0x58

Reset value: 0xXFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFF80 FFFF

(register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 1 WPR first area A unlock
0: WRP1A start and end pages locked
1: WRP1A start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **END[6:0]**: Bank 1 WPR first area A end page
This bitfield contains the last page of the first WPR area in bank 1.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **STRT[6:0]**: Bank 1 WPR first area A start page
This bitfield contains the first page of the first WPR area for bank 1.

7.10.21 FLASH WRP1 area B address register (FLASH_WRP1BR)

Address offset: 0x5C

Reset value: 0xXFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFF80 FFFF

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 1 WPR second area B unlock
0: WRP1B start and end pages locked
1: WRP1B start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **END[6:0]**: Bank 1 WRP second area B end page
This bitfield contains the last page of the second WRP area in bank 1.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **STRT[6:0]**: Bank 1 WRP second area B start page
This bitfield contains the first page of the second WRP area for bank 1.

7.10.22 FLASH secure watermark2 register 1 (FLASH_SECWM2R1)

Address offset: 0x60

Reset value: 0xFFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFFFF FF80

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **SECWM2_END[6:0]**: End page of second secure area
This bitfield contains the last page of the secure area in bank 2.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **SECWM2_STRT[6:0]**: Start page of second secure area
This bitfield contains the first page of the secure area in bank 2.

7.10.23 FLASH secure watermark2 register 2 (FLASH_SECWM2R2)

Address offset: 0x64

Reset value: 0xXXXX FFFF

This register can not be written if OPTLOCK bit is set.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xB480 FFFF

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDP2EN[7:0]								Res.	HDP2_END[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 **HDP2EN[7:0]:** Hide protection second area enable
0xB4: No HDP area 2
others: HDP second area is enabled.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **HDP2_END[6:0]**: End page of hide protection second area
This bitfield contains the last page of the HDP area in bank 2.

Bits 15:0 Reserved, must be kept at reset value.

7.10.24 FLASH WPR2 area A address register (FLASH_WRP2AR)

Address offset: 0x68

Reset value: 0xXFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFF80 FFFF

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 2 WPR first area A unlock
0: WRP2A start and end pages locked
1: WRP2A start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **END[6:0]**: Bank 2 WPR first area A end page
This bitfield contains the last page of the first WRP area in bank 2.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **STRT[6:0]**: Bank 2 WPR first area A start page
This bitfield contains the first page of the first WRP area for bank 2.

7.10.25 FLASH WPR2 area B address register (FLASH_WRP2BR)

Address offset: 0x6C

Reset value: 0xXFXX FFXX

This register can not be written if OPTLOCK bit is set.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

ST production value: 0xFF80 FFFF

(register bits are loaded with values from the flash memory at OBL)

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]						
rw									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 2 WPR second area B unlock
0: WRP2B start and end pages locked
1: WRP2B start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **END[6:0]**: Bank 2 WPR second area B end page
This bitfield contains the last page of the second WRP area in bank 2.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **STRT[6:0]**: Bank 2 WPR second area B start page
This bitfield contains the first page of the second WRP area for bank 2.

7.10.26 FLASH secure block based bank 1 register x (FLASH_SECBB1Rx)

Address offset: 0x80 + 0x4 *(x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

This register is secure. It can be written only by secure access. This register can be protected against unprivileged access (refer to *Table 63: FLASH_SECBByRx registers access when TrustZone is active (TZEN = 1)*).

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECi:** Page secure/nonsecure attribution (i = 31 to 0)

Each bit is used to set one page security attribution in bank 1.

0: Page (32 * ({x} - 1) + {i}) in bank 1 not block-based secure

1: Page (32 * ({x} - 1) + {i}) in bank 1 block-based secure

7.10.27 FLASH secure block based bank 2 register x (FLASH_SECBB2Rx)

Address offset: 0xA0 + 0x4 *(x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

This register is secure. It can be written only by a secure access. This register can be protected against unprivileged access (refer to *Table 63: FLASH_SECBByRx registers access when TrustZone is active (TZEN = 1)*).

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECi:** Page secure/nonsecure attribution (i = 31 to 0)

Each bit is used to set one page security attribution in bank 2.

0: Page (32 * ({x} - 1) + {i}) in bank 2 not block-based secure

1: Page (32 * ({x} - 1) + {i}) in bank 2 block-based secure

7.10.28 FLASH secure HDP control register (FLASH_SECHDPCR)

Address offset: 0xC0

Reset value: 0xA3A3 A3A3

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDP2EXT_ACCDIS[7:0]								HDP1EXT_ACCDIS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HDP2_ACCDIS[7:0]								HDP1_ACCDIS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **HDP2EXT_ACCDIS[7:0]:** HDP2 extension area access disable

Register access depends on its current value. Refer to *HDP extension area access* for the full conditions.

0xA3: Access to HDP2 extension area granted

0x5C: Access to HDP2 extension area and HDP2 area denied, but HDP2_EXT (in FLASH_HDPEXTR) increment allowed at any time. (SECWM2Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*).

Others: Access to HDP2 extension area and HDP2 area denied. Update of HDP2_EXT size is not possible anymore .(SECWM2Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*)

Bits 23:16 **HDP1EXT_ACCDIS[7:0]:** HDP1 extension area access disable

Register access depends on its current value. Refer to *HDP extension area access* for the full conditions.

0xA3: Access to HDP1 extension area granted

0x5C: Access to HDP1 extension area and HDP1 area denied, but HDP1_EXT (in FLASH_HDPEXTR) increment allowed at any time. (SECWM1Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*)

Others: Access to HDP1 extension area and HDP1 area denied. update of HDP1_EXT size is not possible anymore .(SECWM1Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*)

Bits 15:8 **HDP2_ACCDIS[7:0]:** HDP2 area access disable

When set to a value different than 0xA3, this bitfield cannot be modified except by a system reset.

0xA3: Access to HDP2 area granted

Others: Access to HDP2 area denied (SECWM2Ry option-byte modification blocked - refer to *Rules for modifying specific option bytes*). When HDP2_ACCDIS value is different than 0xA3, any write access to HDP2_ACCDIS is ignored.

Bits 7:0 **HDP1_ACCDIS[7:0]:** HDP1 area access disable

When set to a value different than 0xA3, this bitfield cannot be modified except by a system reset.

0xA3: Access to HDP1 area granted

Others: Access to HDP1 area denied (SECWM1Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*). When HDP1_ACCDIS value is different than 0xA3, any write access to HDP1_ACCDIS is ignored.

7.10.29 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0xC4

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. PRIV is a nonsecure bit. SPRIV is a secure bit.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PRIV**: Privileged protection for nonsecure registers

This bit can be read by both privileged or unprivileged, secure and nonsecure access. This bit can be written by a secure or nonsecure privileged access. A secure or nonsecure unprivileged write access on PRIV bit is ignored

0: Nonsecure FLASH registers can be read and written by privileged or unprivileged access.

1: Nonsecure FLASH registers can be read and written by privileged access only.

Bit 0 **SPRIV**: Privileged protection for secure registers

This bit can be accessed only when TrustZone is enabled (TZEN = 1). This bit can be read by both privileged or unprivileged, secure and nonsecure access. This bit can be written only by a secure privileged access. A nonsecure write access on SPRIV bit is ignored. A secure unprivileged write access on SPRIV bit is ignored.

0: Secure FLASH registers can be read and written by privileged or unprivileged access.

1: Secure FLASH registers can be read and written by privileged access only.

7.10.30 FLASH HDP extension register (FLASH_SECHDPEXTR)

Address offset: 0xC8

Reset value: 0x0000 0000

This register is secure. It can be read and written only by secure access. This register can be protected against unprivileged access when SPRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HDP2_EXT[7:0]**: HDP area extension in 4-Kbyte pages in bank 2
The extension starts at page offset HDP2_END+1 if HDP2 area is enable (HDP2EN = 0xB4). Otherwise, it starts at page offset SECWM2_STRT.
Register access depends on HDP2EXT_ACCDIS value (Refer to *HDP extension area access* for the full conditions).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **HDP1_EXT[7:0]**: HDP area extension in 4-Kbyte pages in bank 1
The extension starts at page offset HDP1_END+1 if HDP1 area is enable (HDP1EN = 0xB4). Otherwise, it starts at page offset SECWM1_STRT.
Register access depends on HDP1EXT_ACCDIS value (refer to *HDP extension area access* for the full conditions.

7.10.31 FLASH privilege block based bank 1 register x (FLASH_PRIVBB1Rx)

Address offset: 0xD0 + 0x4 *(x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

This register is privileged. It can be read written only by a privileged access. This register can be protected against nonsecure access (refer to *Table 64: FLASH_PRIVBByRx registers access when TrustZone is active (TZEN = 1)*).

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVi:** Page privileged/unprivileged attribution (i = 31 to 0)
Each bit is used to set one page privilege attribution in bank 1.
0: Page (32 * ({x} - 1) + {i}) in bank 1 accessible by unprivileged access
1: Page (32 * ({x} - 1) + {i}) in bank 1 only accessible by privileged access

7.10.32 FLASH privilege block based bank 2 register x (FLASH_PRIVBB2Rx)

Address offset: 0xF0 + 0x4 *(x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

This register is privilege. It can be read written only by a privileged access. This register can be protected against nonsecure access (refer to *Table 64: FLASH_PRIVBByRx registers access when TrustZone is active (TZEN = 1)*).

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVi:** Page privileged/unprivileged attribution (i = 31 to 0)
Each bit is used to set one page security attribution in bank 2.
0: Page (32 * ({x} - 1) + {i}) in bank 2 accessible by unprivileged access
1: Page (32 * ({x} - 1) + {i}) in bank 2 only accessible by privileged access

7.10.33 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)

Address offset: 0x110

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[31:0]**: OEM1[31:0] bytes key

7.10.34 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)

Address offset: 0x114

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[63:32]:** OEM1[63:32] bytes key

7.10.35 FLASH OEM1 key register 3 (FLASH_OEM1KEYR3)

Address offset: 0x118

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[95:64]**: OEM1[95:64] bytes key

7.10.36 FLASH OEM1 key register 4 (FLASH_OEM1KEYR4)

Address offset: 0x11C

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM1KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[127:96]**: OEM1[127:96] bytes key

7.10.37 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)

Address offset: 0x120

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[31:0]**: OEM2[31:0] bytes key

7.10.38 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)

Address offset: 0x124

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[63:32]:** OEM2[63:32] bytes key

7.10.39 FLASH OEM2 key register 3 (FLASH_OEM2KEYR3)

Address offset: 0x128

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[95:64]**: OEM2[95:64] bytes key

7.10.40 FLASH OEM2 key register 4 (FLASH_OEM2KEYR4)

Address offset: 0x12C

Reset value: 0x0000 0000

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state when no option bytes modification is ongoing; word, half-word ,and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEM2KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[127:96]**: OEM2[127:96] bytes key

7.10.41 FLASH OEM key status register (FLASH_OEMKEYSR)

Address offset: 0x130

Reset value: 0x00XX 00XX

This register is nonsecure. It can be read by both secure and nonsecure access. This register can be protected against unprivileged access when PRIV = 1 in FLASH_PRIVCFGR register.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM2KEYCRC[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OEM1KEYCRC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OEM2KEYCRC[7:0]**: 8-bit OEM2KEY CRC

CRC is automatically computed by hardware while loading the OEM2KEYkey. The CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X_8 + X_2 + X + 1$ (according the convention). Source code is available in *OEM Key CRC source*.

Note: CRC information is updated each time FLASH OEM2KEY registers are written

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **OEM1KEYCRC[7:0]**: 8-bit OEMKEY1 CRC

CRC is automatically computed by hardware while loading the OEM1KEY key. The CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X_8 + X_2 + X + 1$ (according the convention). Source code is available in *OEM Key CRC source.*

Note: CRC information is updated each time FLASH OEM1KEY registers are written

7.10.42 FLASH register map

Table 67. FLASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_PD	PDREQ2	PDREQ1	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value																		0	0	0	0			0					0	0	0	1
0x04	Reserved	Reserved																															
0x08	FLASH_KEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FLASH_SKEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FLASH_OPT-KEYR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	Reserved	Reserved																															
0x18	FLASH_PD-KEY1R	KEY1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FLASH_PD-KEY2R	KEY2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD2	PD1	OEM2LOCK	OEM1LOCK	WDW	BSY	Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value											0	0	X	X	X	X			0						0	0	0	0	0		0	0
0x24	FLASH_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDW	BSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value															0	0									0	0	0	0	0		0	0
0x28	FLASH_CR	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT	MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1	1			0		0	0							0	0	0	0			0		0	0	0	0	0	0	0	0	0	0
0x2C	FLASH_SCR	LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1		0				0	0								0	0	0			0		0	0	0	0	0	0	0	0	0	0

Table 67. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	FLASH_ECCORR	Res.	ECCC	Res.	Res.	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:0]																		
	Reset value		0						0		0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	FLASH_ECCDETR	ECCD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSF_ECC	BK_ECC	Res.	Res.	ADDR_ECC[18:0]																		
	Reset value	0									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	FLASH_OPSR	CODE_OP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SYSF_OP	BK_OP	Res.	Res.	ADDR_OP[18:0]																		
	Reset value	X	X	X							X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	Reserved	Reserved																															
0x40	FLASH_OPTR	TZEN	IO_VDDIO2_HSLV	IO_VDD_HSLV	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1_RST	NRST_SHDW	NRST_STDBY	NRST_STOP	BDRST_POR	BOR_LEV[2:0]			RDP[7:0]							
	Reset value	X	X	X		X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x44	FLASH_BOOT0R	ADD[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
0x48	FLASH_BOOT1R	ADD[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
0x4C	FLASH_SBOOT0R	ADD[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							X
0x50	FLASH_SECWM1R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x54	FLASH_SECWM1R2	HDP1EN[7:0]								Res.	HDP1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X																
0x58	FLASH_WRP1AR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x5C	FLASH_WRP1BR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X

Table 67. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	FLASH_SECWM2R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x64	FLASH_SECWM2R2	HDP2EN[7:0]								Res.	HDP2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X																
0x68	FLASH_WRP2AR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x6C	FLASH_WRP2BR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT[6:0]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x70-0x7C	Reserved	Reserved																															
0x80+ 0x4 * (x - 1), (x=1 to 4)	FLASH_SECB1Rx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90-0x9C	Reserved	Reserved																															
0xA0+ 0x4 * (x - 1), (x=1 to 4)	FLASH_SECBB2Rx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0-0xBC	Reserved	Reserved																															
0xC0	FLASH_SECHD-PCR	HDP2EXT_ACCDIS‘7:0]								HDP1EXT_ACCDIS‘7:0]								HDP2_ACCDIS‘7:0]								HDP1_ACCDIS‘7:0]							
	Reset value	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	1	1
0xC4	FLASH_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SPRIV
	Reset value																															0	0
0xC8	FLASH_SECHD-PEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	.HDP1_EXT[7:0]						
	Reset value									0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0
0xCC	Reserved	Reserved																															
0xD0+ 0x4 * (x - 1), (x=1 to 4)	FLASH_PRIVBB1Rx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE0 to 0xEC	Reserved	Reserved																															
0xF0+ 0x4 * (x - 1), (x=1 to 4)	FLASH_PRIVBB2Rx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100 to 0x10C	Reserved	Reserved																															

Table 67. FLASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x110	FLASH_OEM1KEYR1	OEM1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	FLASH_OEM1KEYR2	OEM1KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118	FLASH_OEM1KEYR3	OEM1KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C	FLASH_OEM1KEYR4	OEM1KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120	FLASH_OEM2KEYR1	OEM2KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124	FLASH_OEM2KEYR2	OEM2KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x128	FLASH_OEM2KEYR3	OEM2KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C	FLASH_OEM2KEYR4	OEM2KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130	FLASH_OEMKEYSR	Res	Res	Res	Res	Res	Res	Res	Res	OEM2KEYCRC[7:0]								Res	Res	Res	Res	Res	Res	Res	Res	OEM1KEYCRC[7:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is introduced on the C-AHB code bus of the Cortex®-M33 processor, to improve performance when fetching instructions and data from internal and external memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

8.3 ICACHE implementation

Table 68. ICACHE features

Feature	ICACHE
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	32 bits
Data size of AHB slow master2 interface	32 bits

8.4 ICACHE functional description

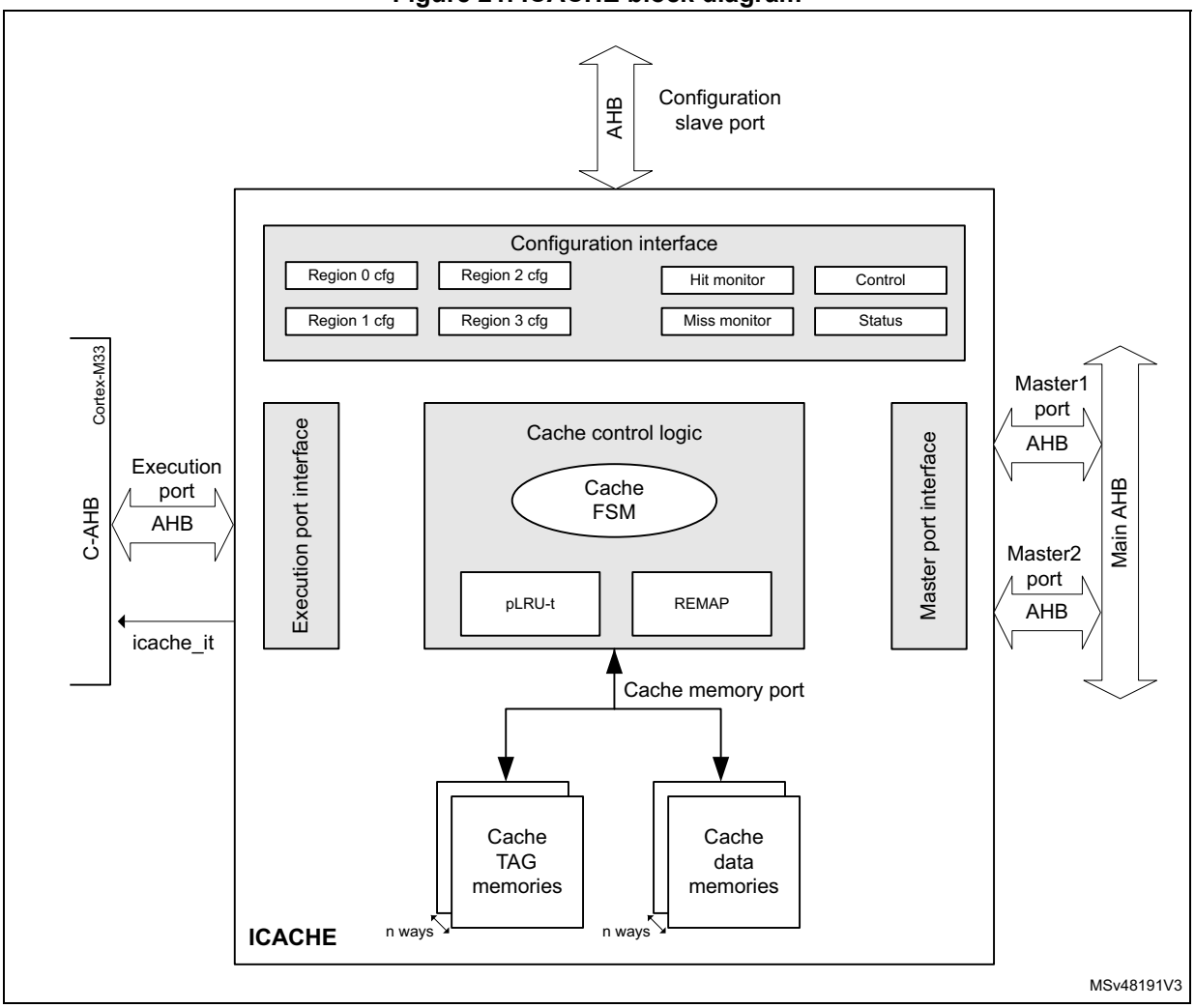
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

8.4.1 ICACHE block diagram

Figure 21. ICACHE block diagram



8.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 69* for B and W definitions):

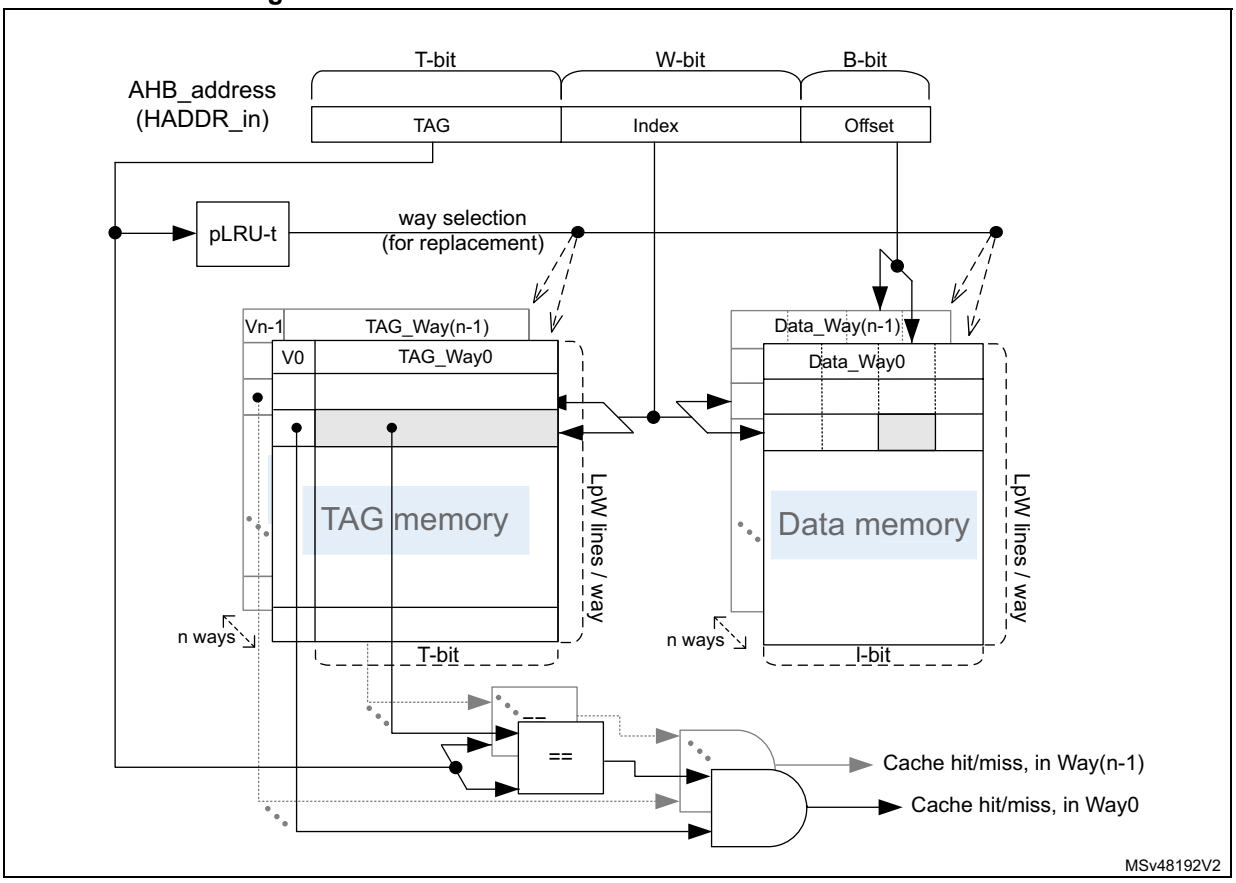
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 22* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 69. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 22. ICACHE TAG and data memories functional view



8.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 70. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 8.4.8*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 23*).

The ICACHE is, by default, disabled at boot.

8.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 8.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 71*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 71. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 8.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 8.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 72 summarizes programmable configurations of various memories.

Table 72. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 23 describes the matching and the output address generation.

Figure 23. ICACHE remapping address mechanism

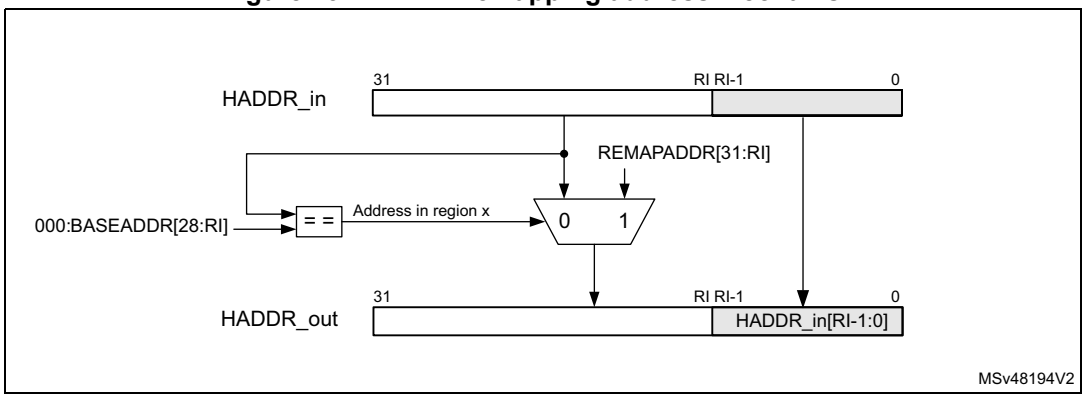


Table 73 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 73. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 73*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.

- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

8.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

8.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

8.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

8.4.12 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 8.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

8.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

8.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 22*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

8.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 74. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the master1 or master2 port back to the execution port.

8.7 ICACHE registers

8.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISS MRST	HITM RST	MISS MEN	HITM EN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAY SEL	CACHE INV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

8.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

8.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

8.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
													w	w	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

8.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

8.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

8.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	MSTSEL	Res.	REMAPADDR[31:21]										
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN	Res.	Res.	Res.	RSIZE[2:0]			Res.	BASEADDR[28:21]							
rw				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

8.7.8 ICACHE register map

Table 75. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Power control (PWR)

9.1 PWR introduction

The power controller manages all device power supplies and power modes transitions.

9.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Backup domain
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage (available on SMPS packages)
  - V_{DDIO2} domain on port PG[15:2]
  - V_{DDUSB} for USB transceiver
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
- Power supply supervision
  - BOR monitor
  - PVD monitor
  - PVM monitor (V_{DDA}, V_{DDUSB}, V_{DDIO2})
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- V_{BAT} battery charging
- TrustZone security and privileged protection

9.3 PWR functional description

9.3.1 PWR pins and internal signals

Table 76. PWR input/output pins

Pin name	Pin type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground

Table 76. PWR input/output pins (continued)

Pin name	Pin type	Description
VDDIO2	Supply	Independent I/O supply
VDDUSB	Supply	USB supply
VDD11 (packages with SMPS)/VCAP (packages without SMPS)	Supply	Logic supply (V_{CORE})
VBAT	Supply	Backup domain supply
VDDSMPS	Supply	SMPS supply
VSSSMPS	Supply	SMPS ground
VLXSMPS	Supply	SMPS output
VREF+	Supply	ADC/DAC high reference voltage
VREF-	Supply	ADC/DAC low reference voltage
WKUPx (x = 1 to 8)	Input	Wake-up pins
PWR_CSLEEP	Output	MCU in Sleep mode
PWR_CSTOP	Output	MCU in Stop mode

Table 77. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx_y (x = 1 to 10, y = 1 to 4)	Input	Wake-up event source selection

Each of ten wake-up events WKUPx, (x = 1 to 10), can be generated from pins or internal events, selected by WUSELx[1:0] in PWR_WUCR3.

Table 78. PWR wake-up source selection

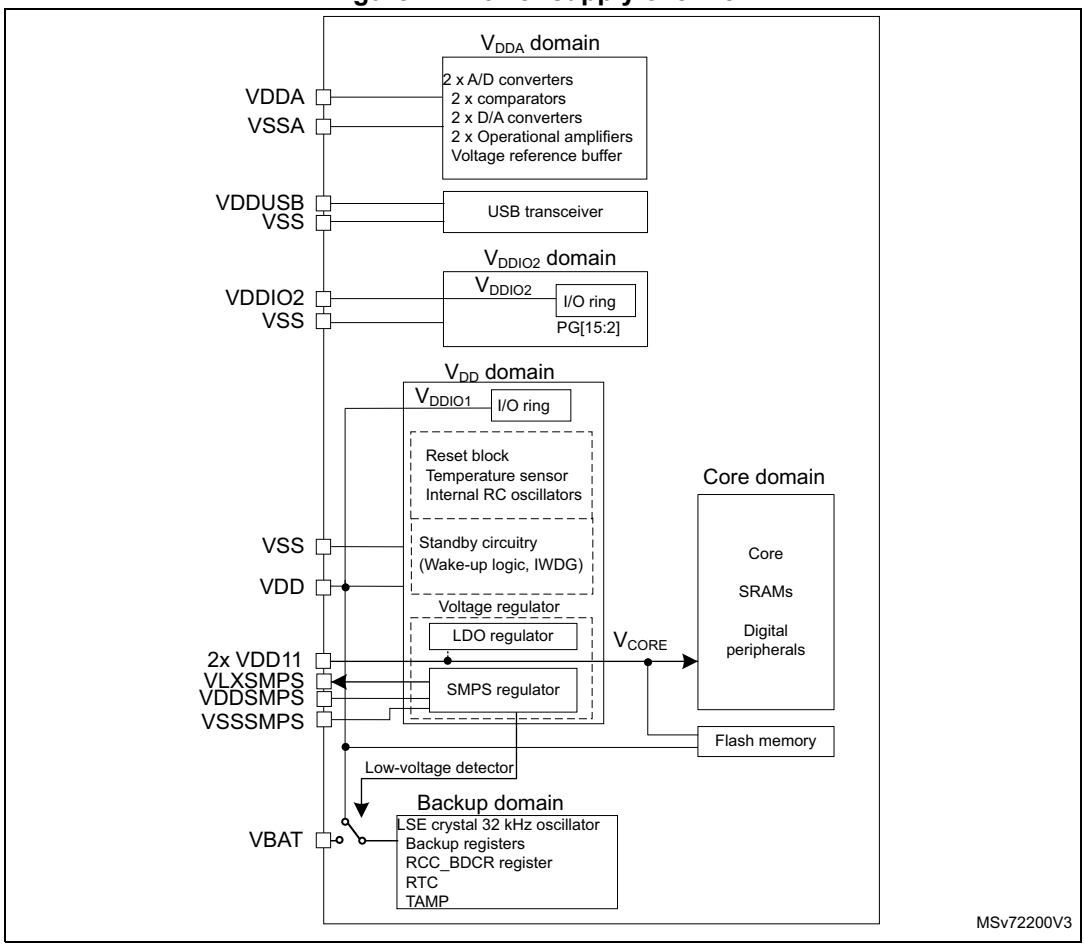
Wake-up event	Internal signal source (x = 1 to 8)			
	WKUPx_0 (WUSELx = 00)	WKUPx_1 (WUSELx = 01)	WKUPx_2 (WUSELx = 10)	WKUPx_3 (WUSELx = 11)
WKUP1	PA0	PB2	PE4	Reserved
WKUP2	PA4	PC13	PE5	Reserved
WKUP3	PE6	PA1	PB6	Reserved
WKUP4	PA2	PB1	PB7	Reserved
WKUP5	PC5	PA3	PB8	IWDG
WKUP6	PB5	PA5	PE7	RTC_ALRA_S or RTC_ALRB_S or RTC_WUT_S or RTC_TS_S
WKUP7	PB15	PA6	PE8	RTC_ALRA or RTC_ALRB or RTC_WUT or RTC_TS
WKUP8	-	PA7	PB10	TAMP

Table 78. PWR wake-up source selection (continued)

Wake-up event	Internal signal source (x = 1 to 8)			
	WKUPx_0 (WUSELx = 00)	WKUPx_1 (WUSELx = 01)	WKUPx_2 (WUSELx = 10)	WKUPx_3 (WUSELx = 11)
WKUP9	I3C1 reset pattern			
WKUP10	I3C2 reset pattern			

9.3.2 PWR power supplies and supply domains

Figure 24. Power supply overview



External power supplies

The devices require a 1.71 V to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals. Those supplies must not be provided without a valid operating supply on the VDD pin:

- V_{DD} = 1.71 V to 3.6 V (functionality guaranteed down to V_{BOR} minimum value)
  V_{DD} is the external power supply for the I/Os, the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through the VDD pins.

- V_{DDA} = 1.58 V (COMPs) / 1.6 V (DACs, OPAMP) / 1.62 (ADCs) / 1.8 V (VREFBUF) to 3.6 V

  V_{DDA} is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, operational amplifiers and comparators. The V_{DDA} voltage level is independent from the V_{DD} voltage and must be connected to VDD when these peripherals are not used.

- V_{DDSMPS} = 1.71 V to 3.6 V

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin, and must be connected to the same supply as VDD pin when the SMPS is used in the application. When the SMPS is not used, it is recommended to connect both VDDSMPS and VLXSMPS to GND.

- V_{LXSMPS} is the SMPS step-down converter output.

  Note: The SMPS power supply pins are available only on a specific package with SMPS step-down converter option.

- V_{DDUSB} = 3.0 V to 3.6 V

  V_{DDUSB} is the external independent power supply for USB transceivers. The V_{DDUSB} voltage level is independent from the V_{DD} voltage and must preferably be connected to VDD when the USB is not used.

- V_{DDIO2} = 1.08 V to 3.6 V

  V_{DDIO2} is the external power supply for 14 I/Os (port G[15:2]). The V_{DDIO2} voltage level is independent from the V_{DD} voltage and must preferably be connected to VDD when PG[15:2] are not used.

- V_{BAT} = 1.65 V to 3.6 V (functionality guaranteed down to V_{BOR_VBAT} minimum value, refer to the product datasheet)

  V_{BAT} is the power supply when V_{DD} is not present (through power switch) for RTC, TAMP, external 32 kHz oscillator, and backup registers.

- V_{REF-}, V_{REF+}

  V_{REF+} is the input reference voltage for ADCs and DACs. It is also the output of the internal voltage reference buffer when enabled.

  V_{REF+} can be grounded when ADC and DAC are not active.

  The internal voltage reference buffer supports four output voltages, that are configured with VRS bit in the VREFBUF_CSR register:

  – V_{REF+} around 1.5 V. This requires $V_{DDA} \geq 1.8$ V.
  – V_{REF+} around 1.8 V. This requires $V_{DDA} \geq 2.1$ V.
  – V_{REF+} around 2.048 V. This requires $V_{DDA} \geq 2.4$ V.
  – V_{REF+} around 2.5 V. This requires $V_{DDA} \geq 2.8$ V.

  VREF- and VREF+ pins are not available on all packages. When not available, they are bonded to VSSA and VDDA, respectively.

  When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled.

  V_{REF-} must always be equal to V_{SSA}.

Internal regulators

The devices embed two regulators: one LDO and one SMPS in parallel to provide the V_{CORE} supply for digital peripherals, SRAMs and embedded flash memory. The SMPS

generates this voltage on VDD11 (two pins) with a total external capacitor of 4.7 µF typical and requires an external coil of 2.2 µH typical.

The LDO generates this voltage on VCAP (one pin) with a total of external capacitor of 4.7 µF typical.

Both regulators can provide two different voltages (voltage scaling) and can operate in Stop mode.

It is possible to switch from SMPS to LDO and from LDO to SMPS on the fly.

Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{DDIO2}, V_{DDUSB}) must remain below V_{DD} + 300 mV.
- When V_{DD} is above 1 V, all power supplies are independent.

During the power-down phase, V_{DD} can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling capacitors to be discharged with different time constants during the power-down transient phase.

Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB:

- The analog peripherals voltage supply input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (A/D converter, D/A converter, comparators, operational amplifiers, voltage reference buffer).

After reset, ADC and analog switch control supplied by V_{DDA} are logically and electrically isolated and therefore are not available. The isolation must be removed before using the analog peripherals, by setting the ASV bit in the PWR_SVMCR register, once the V_{DDA} supply is present.

The V_{DDA} supply can be monitored by the analog voltage monitors (AVM), and compared with two thresholds (1.6 V for AVM1 or 1.8 V for AVM2), refer to *Peripheral voltage monitoring (PVM)* for more details.

When a single supply is used, V_{DDA} can be externally connected to V_{DD} through the external filtering circuit in order to ensure a noise-free V_{DDA} reference voltage.

ADC and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to VREF+, a separate reference voltage lower than V_{DDA}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

V_{REF+} can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF).

The internal voltage reference can output a configurable voltage: 1.5 V, 1.8 V, 2.048 V or 2.4 V. The internal voltage reference can also provide the voltage to external components through VREF+ pin. Refer to the device datasheet and to *Section 24.5.3: VREFBUF register map* for further information.

Independent I/O supply rail

Some I/Os from port G (PG[15:2]) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 V to 3.6 V and is provided externally through the VDDIO2 pin. The V_{DDIO2} voltage level is completely independent from V_{DD} or V_{DDA}. The VDDIO2 pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for I/O list(s).

After reset, the I/Os supplied by V_{DDIO2} are logically and electrically isolated and therefore are not available. The isolation must be removed before using any I/O from PG[15:2], by setting the IO2SV bit in PWR_SVMCR, once the V_{DDIO2} supply is present.

The V_{DDIO2} supply is monitored by the IO2 voltage monitoring (IO2VM) and compared with the internal reference voltage (3/4 VREFINT, around 0.9 V), refer to *Peripheral voltage monitoring (PVM)* for more details.

Independent USB transceivers supply

The USB transceivers are supplied from a separate VDDUSB power supply pin. V_{DDUSB} range is from 3.0 V to 3.6 V and is completely independent from V_{DD} or V_{DDA}. The VDDUSB pin is not available for some small packages. It is internally connected with V_{DD}. Refer to the pinout diagrams or tables in the related device datasheet(s) for I/O list(s).

After reset, the USB features supplied by V_{DDUSB} are logically and electrically isolated and therefore are not available. The isolation must be removed before using the USB peripheral, by setting the USV bit in PWR_SVMCR, once the V_{DDUSB} supply is present.

The V_{DDUSB} supply is monitored by the USB voltage monitoring (UVM) and compared with the internal reference voltage (V_{REFINT}, around 1.2 V), refer to *Peripheral voltage monitoring (PVM)* for more details.

Battery backup domain (also known as RTC domain)

To retain the content of the backup registers and supply the RTC and TAMP functions when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The backup domain supply is V_{SW}, which is the output of a power switch between V_{DD} and V_{BAT}. The switch between V_{DD} and V_{BAT} supplies is automatically controlled by the brownout reset circuitry.

When V_{DD} is below the lowest brownout reset threshold (V_{BOR0}), the VBAT pin powers the RTC and TAMP peripherals, and LSE oscillators.

PC13, PC14 and PC15 pins, that can be configured by the RTC, the TAMP, or the LSE, are also powered by the VBAT pin.

When V_{DD} is higher than V_{BOR0}, the V_{DD} pin powers all previous functions.

Note: Due to the fact that the analog power switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed must be limited (refer to the datasheet for more details) and these I/Os must not be used as a current source (for example to drive a LED).

Caution: During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR has been detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.

Caution: During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}). If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor.

Backup domain access

After a system reset, the backup domain (RCC_BDCR, RTC registers, TAMP registers and backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits in the RCC_AHB1ENR2.
2. Set the DBP bit in the PWR disable backup domain register (PWR_DBPR) to enable access to the backup domain.

VBAT battery charging

When V_{DD} is present, It is possible to charge the external battery on VBAT through an internal resistance.

The V_{BAT} charging is done either through a 5 kΩ resistor or through a 1.5 kΩ resistor depending on the VBRS bit value in PWR_BDCR.

The battery charging is enabled by setting VBE bit in PWR_BDCR. It is automatically disabled in V_{BAT} mode.

9.3.3 PWR system supply voltage regulation

SMPS and LDO embedded regulators

The devices embed two internal regulators, that can be selected when the application runs, depending on the application requirements:

- a SMPS step-down converter
- a linear voltage regulator (LDO)

The SMPS allows the power consumption to be reduced but some applications can be perturbed by the noise generated by the SMPS, requiring the application to switch to LDO.

The LDO and the SMPS regulators have two modes: Main regulator mode (used when performance is needed), and Low-power regulator mode. LDO or SMPS can be used in all voltage scaling ranges, and in all Stop modes.

LDO and SMPS versus reset, voltage scaling, and low-power modes

After reset, the regulator is the LDO, in range 2. Switching to SMPS provides lower consumption in particular at high V_{DD} voltage. It is possible to switch from LDO to SMPS, or from SMPS to LDO in any range, by configuring the REGSEL bit.

When exiting the Stop or Standby mode, the regulator is the same than when entering low-power modes. The voltage range is the range 2, except when exiting Stop 0 mode.

LDO and SMPS step-down converter fast startup

Note: After BOR reset, the LDO and SMPS regulators starts in slowstartup mode. This slowstartup feature is selected to limit the inrush current after power-on reset. This increases the wake-up time when exiting Stop or Standby mode.

However, it is possible to configure a faster startup on the fly and it is applied for next startup either after a system reset or wake-up from low-power mode except Shutdown and V_{BAT} modes. The fast startup is selected by setting the FSTEN bit in the PWR_CR3 register.

Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique that consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V_{CORE} is known as overvolting. It allows the device to improve its performance.

Dynamic voltage scaling to decrease V_{CORE} is known as undervolting. It is performed to save power, particularly in laptop and other mobile devices where the energy comes from a battery and is thus limited.

The regulator operates in the following ranges:

- Range 1: high performance

  It provides a typical output voltage at 0.9 V. It is used when the system clock frequency is up to 96 MHz.
- Range 2: low-power range

  It provides a typical output voltage at 0.75 V. The system clock frequency can be up to 48 MHz.

Voltage scaling is selected through the R1EN and R2EN bits in the PWR_VOSR register. The EPOD (embedded power distribution) booster must be enabled and ready before increasing the system clock frequency above 24 MHz in range 1 and range 2.

The sequence to switch the voltage scaling from range 2 (lower power) to range 1 (high performance) with system clock above 24 MHz is the following:

1. Configure the BOOSTSEL[1:0] and the BOOSTDIV[3:0] in the RCC_CFGR4 register to generate a booster clock frequency between 3 and 16 MHz.
2. Program R1EN = 1 and R2EN = 0 in the PWR_VOSR register.
3. Wait until the R1RDY flag is set in the PWR_VOSR register.
4. Set BOOSTEN in the PWR_VOSR register. This step can be done together with R1EN and R2EN programming
5. Wait until the BOOSTRDY flag is set in the PWR_VOSR register.
6. Adjust number of wait states according new frequency target in range 1 (LATENCY bits in the FLASH_ACR).
7. Configure and switch to new system frequency.

The sequence to switch the voltage scaling from range 1 (higher performance) to range 2 (lower power) is:

1. Reduce the system frequency to a value lower than 48 MHz.
2. Adjust number of wait states according new frequency target (LATENCY bits in the FLASH_ACR).
3. If new SYSCLK ≤ 24 MHz, clear BOOSTEN in the PWR_VOSR register.
4. Program R1EN = 0 and R2EN = 1 in the PWR_VOSR register. This step can be done together with BOOSTEN clearing.

Caution: The booster clock source must not be disabled when the booster is enabled.

9.3.4 PWR power supply supervision

Brownout reset (BOR)

The device has an integrated brownout reset (BOR) circuitry. The BOR is active in all power modes except Shutdown mode, and cannot be disabled.

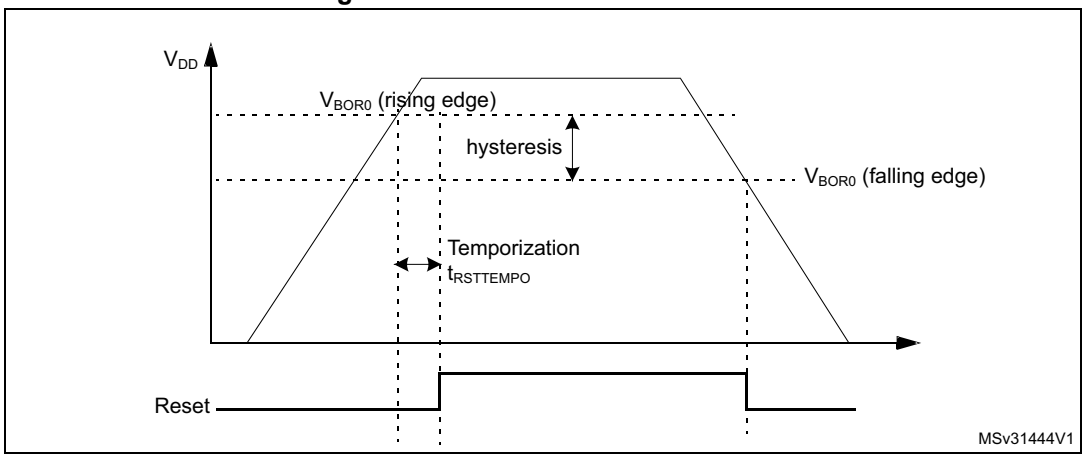
Five BOR thresholds can be selected through option bytes. BOR0 provides the always enabled power-on/powerdown functionality, independent from any other higher BOR level selection.

During power-on, the BOR keeps the device under reset until the supply voltage V_{DD} reaches the specified VBORx threshold. When V_{DD} drops below the selected threshold, a device reset is generated. When V_{DD} is above the VBORx upper limit, the device reset is released and the system can start.

For more details on the brownout reset thresholds, refer to the electrical characteristics section in the datasheet.

During Standby mode and if BOR level 0 is selected, it is possible to set the BOR in ultra-low-power mode to further reduce the current consumption by setting the ULPMEN bit in PWR control register 1 (PWR_CR1).

Figure 25. Brownout reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (VBOR0).

Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PVDLS[2:0] bits in the PWR supply voltage monitoring control register (PWR_SVMCR).

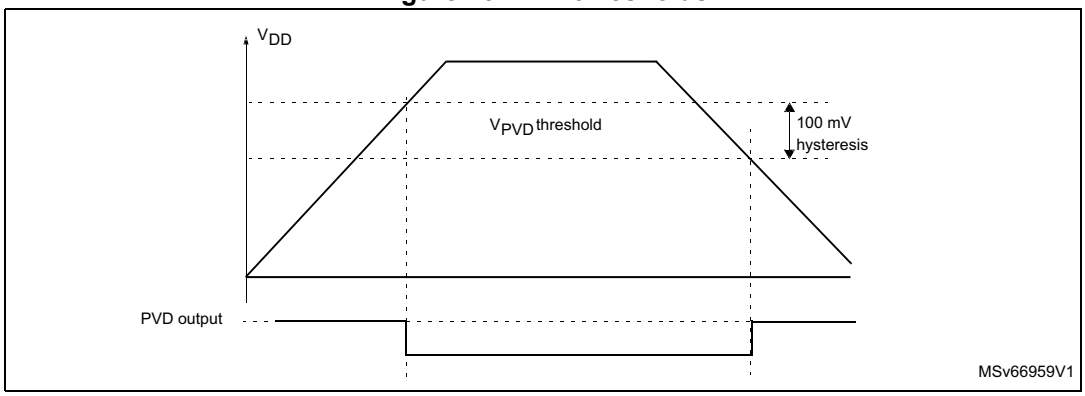
The PVD is enabled by setting the PVDE bit.

A PVDO flag is available in the PWR supply voltage monitoring control register (PWR_SVMCR) to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers (refer to *Table 91: PWR interrupt requests*).

The rising/falling edge sensitivity of the EXTI Line must be configured according to PVD output behavior. For example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_{DD} drops below the PVD threshold. As an example, the service routine can perform emergency shutdown tasks.

The PVD can remain active in Stop 0, Stop 1, Stop 2 modes, and the PVM interrupt can wake up from the Stop mode. The PVD is not functional in Stop 3 mode.

Figure 26. PVD thresholds



Peripheral voltage monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The other supplies (V_{DDA}, V_{DDIO2} and V_{DDUSB}) can be independent from V_{DD} and can be monitored with four peripheral voltage monitoring (PVM):

- The UVM monitors the USB supply V_{DDUSB}. VDDUSBRDY indicates if the V_{DDUSB} independent power supply is higher or lower than the V_{UVM} threshold.
- The IO2VM monitors the PG[15:2] supply V_{DDIO2}. VDDIO2RDY indicates if the V_{DDIO2} independent power supply is higher or lower than the VI_{O2VM} threshold.
- The AVM1 monitors the analog supply V_{DDA}. VDDA1RDY indicates if the V_{DDA} independent power supply is higher or lower than the V_{AVM1} threshold.
- The AVM2 monitors the analog supply V_{DDA}. VDDA2RDY indicates if the V_{DDA} independent power supply is higher or lower than the V_{AVM2} threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The PVMx output interrupt is generated when the independent power supply drops below the PVM threshold and/or when it rises above the PVM threshold, depending on EXTI line rising/falling edge configuration.

Refer to *Table 91: PWR interrupt requests*.

Each PVM can remain active in Stop 0, Stop 1, Stop 2 modes, and the PVM interrupt can wake up from the Stop mode. The PVM is not functional in Stop 3 mode.

The independent supplies (V_{DDA}, V_{DDIO2} and V_{DDUSB}) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies:

Table 79. PVM features

PVM	Power supply	PVM threshold
UVM	V_{DDUSB}	V_{UVM} (around 1.2 V)
IO2VM	V_{DDIO2}	V_{IO2VM} (around 0.9 V)
AVM1	V_{DDA}	V_{AVM1} (around 1.6 V)
AVM2	V_{DDA}	V_{AVM2} (around 1.8 V)

- If these supplies are shorted externally to V_{DD}, the application assumes they are available without enabling any peripheral voltage monitoring.
- If these supplies are independent from V_{DD}, the peripheral voltage monitoring (PVM) can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB peripheral:

1. If V_{DDUSB} is independent from V_{DD}:
   a) Enable the UVM by setting UVMEN bit in the PWR supply voltage monitoring control register (PWR_SVMCR).
   b) Wait for the UVM wake-up time.
   c) Wait until VDDUSBRDY bit is set in the PWR supply voltage monitoring status register (PWR_SVMSR).
   d) Disable the UVM for consumption saving (optional).
2. Set the USV bit in the PWR supply voltage monitoring control register (PWR_SVMCR) to remove the V_{DDUSB} power isolation.

The following sequence must be done before using any I/O from PG[15:2]:

1. If V_{DDIO2} is independent from V_{DD}:
   a) Enable the IO2VM by setting IO2VM bit in the PWR supply voltage monitoring control register (PWR_SVMCR).
   b) Wait for the IO2CVM wake-up time.
   c) Wait until VDDIO2RDY bit is set in the PWR supply voltage monitoring status register (PWR_SVMSR).
   d) Disable the IO2VM for consumption saving (optional).
2. Set the IO2SV bit in the PWR supply voltage monitoring control register (PWR_SVMCR) to remove the V_{DDIO2} power isolation.

The following sequence must be done before using any of these analog peripherals: analog to digital converters, digital to analog converters, comparators, operational amplifiers, voltage reference buffer:

1. If V_{DDA} is independent from V_{DD}:
   a) Enable the AVM1 or AVM2 by setting AVM1EN or AVM2EN bit in the PWR supply voltage monitoring control register (PWR_SVMCR).
   b) Wait for the AVM wake-up time.
   c) Wait until VDDA1RDY or VDDA2RDY bit is set in the PWR supply voltage monitoring status register (PWR_SVMSR).
   d) Disable the AVM for consumption saving (optional).
2. Set the ASV bit in the PWR supply voltage monitoring control register (PWR_SVMCR) to remove the V_{DDA} power isolation.

9.3.5 PWR power management

Power modes overview

By default, the microcontroller is in Run mode after a system or a power reset. Reducing power consumption in Run mode is done by configuring voltage scaling according to application performance needs. Refer to *Dynamic voltage scaling management*. Unused RAMs can definitively be powered-off with SRAM1PD or SRAM2PD bits in PWR_CR1. Once an SRAM has been powered off using this configuration bit, it cannot be powered on back again until the next power-on reset. Power consumption is also reduced by reducing SYSCLK, HCLK and PCLK clocks speed, or gating unused peripherals clocks. Refer to *Section 10: Reset and clock control (RCC)* for more details.

Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wake-up sources.

The device features these low-power modes:

- Sleep mode:

  CPU clock off, all peripherals including Cortex-M33 core such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs (refer to *Sleep mode*).
- Stop 0, Stop 1, Stop 2, Stop 3 modes:

Stop mode achieves the lowest power consumption while retaining the content of SRAM and registers. In Stop 0 and Stop 1, the SRAMs can be programmed, with the SRAMFWU bit of the PWR_CR2 register, to remain in normal run mode, to allow short wake-up time.

In Stop 1, Stop 2, and Stop3, the SRAMs can be totally or partially switched off, to further reduce consumption (refer to the xRAMyPDS bits of the PWR_CR2 register). When the SRAM is definitively powered down through PWR_CR1, the power-down in Stop mode configuration in the PWR_CR2 register for the same SRAM has no effect. All clocks in the core domain are stopped. The MSI (MSIS and MSIK) RC, the HSI16 RC and the HSE crystal oscillators are disabled. The LSE or LSI is still running.

The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

Some peripherals are autonomous and can operate in Stop mode by requesting their kernel clock and their bus (APB or AHB) when needed, in order to transfer data with GPDMA1 depending on peripherals and power mode.

In Stop 0 mode, the regulator remains in main regulator mode, allowing a very fast wake-up time but with much higher consumption.

In Stop 1, the regulator is in low power mode, and the whole core domain is fully powered. All autonomous peripherals are functional.

In Stop 2 mode, most of the core domain (D1 domain) is put in a lower leakage mode, keeping registers retention, but without any possible functionality. The D2 domain, embedding APB3 peripherals, is kept fully powered, so those peripherals can be kept functional.

Stop 3 is the lowest power mode with full retention, but the functional peripherals and sources of wake-up are reduced to the same ones than in Standby mode.

The system clock when exiting from Stop mode can be either MSIS up to 48 MHz or HSI16, depending on software configuration.

In case the wake-up clock is MSIS at 48 MHz, the EPOD booster must be configured (BOOSTSEL[1:0] = 01 in RCC_CFGR4) and enabled (BOOSTEN = 1 in PWR_VOSR) before entering Stop mode. Refer to System clock (SYSCLK) selection for more details.

Refer to *Stop 0 mode*, *Stop 1 mode*, *Stop 2 mode*, and *Stop 3 mode*.

- Standby mode:

  The Standby mode is used to achieve the lowest power consumption with BOR. The internal regulator is switched off so that the core domain is powered off. The MSI (MSIS and MSIK) RC, the HSI16 RC and the HSE crystal oscillators are also switched off.

  The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

  The brownout reset (BOR) always remains active in Standby mode.

  The state of each I/O during Standby mode can be selected by software: I/O with internal pull-up, internal pulldown or floating.

  After entering Standby mode, SRAMs and register contents are lost except for registers in the backup domain and Standby circuitry. Optionally, the full SRAM2 or 8 Kbytes or 24 Kbytes or 32 Kbytes can be retained in Standby mode, supplied by the low-power regulator (standby with SRAM2 retention mode).

  The BOR can be configured in ultra-low-power mode to further reduce power consumption during standby mode and when the lowest threshold is selected (VBOR0).

  The device exits Standby mode when an external reset (NRST pin), an IWDG early wake-up event or reset, WKUP pin event (configurable rising or falling edge), a RTC event occurs (alarm, periodic wake-up, timestamp), a tamper detection or a I3C reset pattern detection.

  The system clock after wake-up is MSIS up to 12 MHz.

  Refer to *Standby mode*.

- Shutdown mode:

  The Shutdown mode allows the lowest power consumption. The internal regulator is switched off so that the core domain is powered off. The HSI16, the MSI (MSIS and MSIK), the LSI and the HSE oscillators are also switched off.

  The RTC can remain active (Shutdown mode with RTC, Shutdown mode without RTC).

  The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to backup domain is not supported.

  SRAMs and register contents are lost except for registers in the backup domain.

  The device exits Shutdown mode when an external reset (NRST pin), a WKUP pin event (configurable rising or falling edge), or a RTC event occurs (alarm, periodic wake-up, timestamp), a tamper detection or a I3C reset pattern detection.

  The system clock after wake-up is MSIS at 12 MHz.

  Refer to *Shutdown mode*.

The table below shows the power modes overview.

Table 80. Low-power mode summary

<table>
<tr><th>Mode name</th><th>Entry</th><th>Wake-up source(1)</th><th>Wake-up system clock</th><th>Effect on clocks</th><th>Voltage regulators</th></tr>
<tr><td rowspan="2">Sleep
(Sleep-now or Sleep-on-exit)</td><td>WFI or Return from ISR</td><td>Any interrupt</td><td rowspan="2">Same as before entering Sleep mode</td><td rowspan="2">CPU clock OFF
No effect on other clocks or analog clock sources</td><td rowspan="2">Range 1, 2</td></tr>
<tr><td>WFE</td><td>Wake-up event</td></tr>
<tr><td>Stop 0</td><td>LPMS = 000 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td rowspan="4">Any EXTI line (configured in the EXTI registers)
Any PWR wake-up line (WKUP pins and I3C reset pattern)
Specific peripherals events/interrupts(2)</td><td rowspan="5">HSI16 when STOPWUCK = 1 in RCC_CFGR1
MSIS with the frequency before entering the Stop mode, limited to 48 MHz, when STOPWUCK = 0</td><td rowspan="4">All clocks OFF except LSI and LSE
MSIK, MSIS or HSI16 can be enabled temporarily when requested by an autonomous peripheral, or forced to be kept enabled.</td><td>Range 1, 2</td></tr>
<tr><td>Stop 1</td><td>LPMS = 001 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td rowspan="4">Low-power regulator (SMPS or LDO)</td></tr>
<tr><td>Stop 2</td><td>LPMS = 010 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td></tr>
<tr><td>Stop 3</td><td>LPMS = 011 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td rowspan="4">All clocks OFF except LSI and LSE</td></tr>
<tr><td>Standby with SRAM2_ 8 Kbytes</td><td>LPMS = 10X+ RRS1 = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td rowspan="3">WKUP pin edge, RTC/TAMP events/interrupts(2), external reset in NRST pin, IWDG events/interrupts(2) or reset, I3C reset pattern</td><td rowspan="3">MSIS from 3 MHz up to 12 MHz</td></tr>
<tr><td>Standby with SRAM2_Full</td><td>LPMS = 10X+ RRS1 = RRS2 = RRS3 = 1+ SLEEPDEEP bit + WFI or Return from ISR or WFE</td></tr>
<tr><td>Standby</td><td>LPMS = 10X + RRS1 = RRS2 = RRS3 = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td>OFF</td></tr>
<tr><td>Shutdown</td><td>LPMS = 11x + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td>WKUP pin edge, RTC/TAMP events/interrupts(2), external reset in NRST pin,I3C reset pattern</td><td>MSIS 12 MHz</td><td>All clocks OFF except LSE</td><td>OFF</td></tr>
</table>

1. Refer to the next table.

2. A wake-up event from Stop mode can be generated with the peripheral interrupt signal. Refer to *Exiting a low-power mode*.

Table 81. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
CPU	Y	-	-	-	-	-	-	-	-	-	-	-	-
Flash memory (1 Mbyte)	O(2)	O(2)	-	O(3)	-	O(3)	-	-	-	-	-	-	-
SRAM1 (192 Kbytes)	Y(4)	Y(5)	O(6)	-	O(6)	-	O(6)	-	-	-	-	-	-
SRAM2 (64 Kbytes)	Y(4)	Y(5)	O(6)	O(7)	O(6)	O(7)	O(6)	-	O(8)	-	-	-	-
OCTOSPI1	O	O	-	-	-	-	-	-	-	-	-	-	-
Backup registers	Y	Y	Y	-	Y	-	Y	-	Y	-	Y	-	Y
Brownout reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
Programmable voltage detector (PVD)	O	O	O	O	O	O	-	-	-	-	-	-	-
Peripheral voltage monitor	O	O	O	O	O	O	-	-	-	-	-	-	-
GTZC	O	O	O	O(9)	O	O(9)	-	-	-	-	-	-	-
RAMCFG	O	O	O	O(7)	O	O(7)	-	-	-	-	-	-	-
GPDMA1	O	O	O(10)	O(10)	O(11)	O(11)	-	-	-	-	-	-	-
High-speed internal (HSI16)	O	O	(12)	-	(12)	-	-	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-	-	-
High-speed external (HSE)	O	O	-	-	-	-	-	-	-	-	-	-	-
Low-speed internal (LSI)	O	O	O	-	O	-	O	-	O	-	-	-	-
Low-speed external (LSE)	O	O	O	-	O	-	O	-	O	-	O	-	O
Multi-speed internal (MSIS and MSIK)	O	O	(12)	-	(12)	-	-	-	-	-	-	-	-
Clock security system (CSS)	O	O	-	-	-	-	-	-	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O	O	O	O	O	O	O	O	O
RTC/TAMP	O	O	O	O	O	O	O	O	O	O	O	O	O
Number of TAMP tamper pins	5	5	5	O	5	O	5	O	5	O	5	O	5
USB	O	O	-	O	-	-	-	-	-	-	-	-	-
USARTx (x=1,3,4,5)	O	O	O(13)	O(13)	-	-	-	-	-	-	-	-	-

Table 81. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
Low-power UART (LPUART1)	O	O	O[13]	O[13]	O[13]	O[13]	-	-	-	-	-	-	-
I2Cx (x = 1,2,)	O	O	O[14]	O[14]	-	-	-	-	-	-	-	-	-
I2C3	O	O	O[14]	O[14]	O[14]	O[14]	-	-	-	-	-	-	-
I3Cx (x=1,2)	O	O	O[15]	O[15]	-	O[16]	-	O[16]	-	O[16]	-	O[16]	-
SPIx (x = 1,2,3,)	O	O	O[17]	O[17]	-	-	-	-	-	-	-	-	-
FDCAN1	O	O	-	-	-	-	-	-	-	-	-	-	-
SDMMC1	O	O	-	-	-	-	-	-	-	-	-	-	-
SAI1	O	O	-	-	-	-	-	-	-	-	-	-	-
ADC12	O	O	-	-	-	-	-	-	-	-	-	-	-
DAC1 (2 converters)	O	O	O[18]	-	-	-	-	-	-	-	-	-	-
VREFBUF	O	O	O	-	-	-	-	-	-	-	-	-	-
OPAMPx (x = 1,2)	O	O	O	-	-	-	-	-	-	-	-	-	-
COMPx (x = 1,2)	O	O	O	O	O	O	-	-	-	-	-	-	-
Temperature sensor	O	O	O	-	-	-	-	-	-	-	-	-	-
Timers (TIMx)	O	O	-	-	-	-	-	-	-	-	-	-	-
LPTIMx (x = 1,3,4)	O	O	O[19]	O[19]	O[19]	O[19]	-	-	-	-	-	-	-
LPTIM2	O	O	O[19]	O[19]	-	-	-	-	-	-	-	-	-
Independent watchdog (IWDG)	O	O	O	O	O	O	O	O	O	O	-	-	-
Window watchdog (WWDG)	O	O	-	-	-	-	-	-	-	-	-	-	-
SysTick timer	O	O	-	-	-	-	-	-	-	-	-	-	-
Audio digital filter (ADF)	O	O	O[20]	O[20]	-	-	-	-	-	-	-	-	-
Touch sensing controller (TSC)	O	O	-	-	-	-	-	-	-	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-	-	-	-	-	-	-	-
AES and secure AES	O	O	-	-	-	-	-	-	-	-	-	-	-
Public key accelerator (PKA)	O	O	-	-	-	-	-	-	-	-	-	-	-
HASH accelerator	O	O	-	-	-	-	-	-	-	-	-	-	-

Table 81. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
CRC calculation unit	O	O	-	-	-	-	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	(21)	22	(21)	22	(22)	22	-

1. Y = yes (enable). O = optional (disable by default, can be enabled by software). - = not available.
2. The flash banks can be configured in power-down mode. By default, they are not in power-down mode.
3. Flash can be accessed by GPDMA1 in Stop 0, Stop 1 and Stop 2 modes. ECC error interrupt or NMI wakes up from these Stop modes.
4. The SRAMs can definitively be powered off independently until the next device power on reset.
5. The SRAM clock can be gated on or off independently.
6. Sub-blocks of SRAM1 and SRAM2 can be powered-off in Stop 1, Stop 2, and Stop 3, to save power. SRAM1 and SRAM2 can be accessed by GPDMA1 in Stop 0, Stop 1 and Stop 2 modes.
7. Parity error interrupt or NMI wakes up from Stop mode.
8. Content of 32 Kbyte, 24 Kbyte, and/or 8 Kbyte block can be preserved.
9. Illegal access interrupt wakes up from Stop 0, 1, 2 modes.
10. GPDMA1 transfers are functional and autonomous in Stop 0 and 1 mode. Interrupts wake up from these Stop modes.
11. In Stop 2 mode, GPDMA1 supports only LPUART1, I2C3, LPTIM1, LPTIM3 and LPTIM4 requests. None of GPDMA1 triggers are supported. Interrupts wake up from Stop 2 modes.
12. Some peripherals with autonomous mode and wake-up from Stop capability can request HSI16, MSIS or MSIK to be enabled. In this case, the oscillator is woken up by the peripheral, and is automatically put off when no peripheral needs it.
13. USART and LPUART reception and transmission is functional and autonomous in Stop mode, in asynchronous and in SPI master modes. Interrupts wake up from Stop mode.
14. I2C reception and transmission is functional and autonomous in Stop mode. Interrupts wake up from Stop mode.
15. I3C reception and transmission, in controller and target modes, is functional and autonomous in Stop mode. Interrupts wake up from Stop mode.
16. I3C reset pattern detection wakes up from Stop 2, Stop 3, Standby and Shutdown modes. I3C pull-ups can be applied in Stop and Standby modes.
17. SPI reception and transmission is functional and autonomous in Stop mode. Interrupts wake up from Stop mode.
18. DAC1 (2 channels) conversion in sample and hold mode is functional and autonomous in Stop mode.
19. LPTIM is functional and autonomous in Stop mode.Interrupts wake up from Stop mode.
20. ADF is functional and autonomous in Stop mode. Interrupts wake up from Stop mode.
21. I/Os can be configured with internal pull-up, pull-down or floating in Stop 3 and Standby mode.
22. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting the Shutdown mode.

Autonomous peripherals

Several peripherals support the autonomous mode which allows it to be functional and perform DMA transfers in Stop 0, Stop 1, and Stop 2 modes. Their interrupts wake up from Stop mode.

In Stop 0 and Stop 1 modes, the autonomous peripherals are DAC1 (2 channels), LPTIMx (x = 1 to 4), U(S)ARTx (x = 1, 3 to 5), LPUART1, SPIx (x = 1 to 3), I2Cx (x = 1 to 3), I3Cx (x = 1 to 2), ADF1, and GPDMA1.

In Stop 2 mode, the autonomous peripherals are LPTIM1, LPTIM3, LPTIM4, LPUART1, and I2C3. If one of these peripherals requests the AHB/APB clocks for a DMA transfer, the whole core domain is switched to Stop 1 higher leakage mode and the clock is distributed to GPDMA1, enabled SRAMs and peripherals in order to perform the autonomous peripheral DMA transfer. Then the core domain automatically returns to Stop 2 lower leakage mode.

Note: Only DMA requests from LPTIM1, LPTIM3, LPTIM4, LPUART1, or I2C3 peripherals are supported in Stop 2 mode. It is not possible to trig a GPDMA1 transfer in Stop 2 mode using GPDMA1 hardware triggers signals, even from those peripherals.

Low-power modes

Entering a low-power mode

The MCU enters low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex-M33 system control register is set on Return from ISR.

Entering a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Caution: The peripherals with autonomous mode feature are able to generate an AHB or APB clock request, depending on their internal events. If a clock request is present when WFI or WFE is executed, the low-power mode entry is delayed until the clock request is released.

Exiting a low-power mode

The way the MCU exits the Sleep or Stop mode depends on the way the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - an NVIC IRQ peripheral interrupt:
    When SEVONPEND = 0 in the Cortex-M33 system control register: By enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC IRQ peripheral channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with high enough priority wake up and interrupt the MCU.
    When SEVONPEND = 1 in the Cortex-M33 system control register: By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC IRQ peripheral channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority wake up and interrupt the MCU.
  - an event: by configuring a EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the

NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It may be necessary to clear the interrupt flag in the peripheral.

After waking up from Standby or Shutdown mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

Caution: When the device is in Stop mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with clock ON.

Caution: The peripherals with autonomous mode feature are able to generate an AHB or APB clock request when the device is in Stop mode, depending on their internal events. The software must ensure that either DMA transfer or interrupt is served, by configuring properly and in a consistent way the RCC, the autonomous peripherals, the DMA channels and NVIC. Note that when an autonomous peripheral requests the bus clock in Stop mode, the AHB and APB clocks are distributed to all enabled peripherals. Consequently, enabled peripherals, even without autonomous mode capability, are temporarily clocked and can also generate an interrupt during this time. These peripherals interrupts wake up the device from Stop mode.

Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering Sleep mode

The MCU enters the Sleep mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

Exiting Sleep mode

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).

Table 82. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.

Table 82. Sleep mode (continued)

Sleep mode	Description
Mode exit	If WFI or Return from ISR was used for entry Interrupt (refer to *Table 128: STM32U3 series vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (refer to *Section 17.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (refer to *Table 128: STM32U3 series vector table*) or wake-up event (refer to *Section 17.3: EXTI functional description*)
Wake-up latency	None

Stop 0 mode

The Stop 0 mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. The MSIS, MSIK, HSI16 and HSE oscillators are disabled by hardware. The LSE and LSI are kept enabled if they were before entering Stop 0 mode.

Some peripherals with the autonomous mode capability can switch on HSI16 or MSIS or MSIK for transferring data (see *Autonomous peripherals* for details).

All SRAMs and register contents are preserved. SRAMs can be kept in normal mode to allow fast wake-up at a cost of extra power consumption.

The BOR is always available in Stop 0 mode.

I/O states in Stop 0 mode

In the Stop 0 mode, all I/O pins keep the same state as in the Run mode.

Entering Stop 0 mode

The MCU enters the Stop 0 mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 83* for details on how to enter the Stop 0 mode).

If the flash memory programming is ongoing, the Stop 0 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 0 mode entry is delayed until the APB access is finished.

In Stop 0 mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it cannot be stopped except by a reset (see Section 10.3 PWR functional description).
- The real-time clock (RTC) and tamper (TAMP) kernel clock is enabled by the RTCEN bit in RCC_BDCR.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_CSR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.

Several peripherals can be autonomous in Stop 0 mode and can add consumption if they are enabled (see *Autonomous peripherals* for more details).

The OPAMPs, the COMPs, the DAC1 (two channels), the VREFBUF, the PVM, and the PVD can be used in Stop 0 mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 2) and the temperature sensor can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exiting Stop 0 mode

The MCU exits the Stop 0 mode as described in *Exiting a low-power mode* (see *Table 83* for details on how to exit Stop 0 mode).

When exiting Stop 0 mode by issuing an interrupt or a wake-up event, HSI16 is selected as system clock if the bit STOPWUCK is set in RCC_CFGR1. The MSIS oscillator is selected as system clock if STOPWUCK is cleared. The MSIS selection allows a wake-up at higher frequency (up to 48 MHz).

Several peripherals are autonomous in Stop mode, and can generate interrupts with wake-up from Stop capability.

All peripheral clocks must be enabled to allow a wake-up from Stop interrupt (Refer to RCC for more details).

When exiting the Stop 0 mode, the MCU is in Run mode (Range 1 or Range 2 depending on regulator range configuration).

Table 83. Stop 0 mode

Stop 0 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 000 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 000 in PWR_CR1
	To enter Stop 0 mode, all EXTI line pending bits and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and the program execution continues.

Table 83. Stop 0 mode (continued)

Stop 0 mode	Description
Mode exit	If WFI or Return from ISR was used for entry: – any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: – any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*). – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt ([PERIPH]EN, [PERIPH]SLPEN and [PERIPH]STPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	Wake-up on flash: – HSI as SYSCLK: HSI setup + flash setup – MSI as SYSCLK: HSI setup + flash setup + MSI setup

Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the regulator is in low-power mode and the SRAMs can be totally or partially switched off to further reduce consumption (see the following table for details on how to enter and exit Stop 1 mode).

Table 84. Stop 1 mode

Stop 0 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 000 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 000 in PWR_CR1
	To enter Stop 0 mode, all EXTI line pending bits and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and the program execution continues.
Mode exit	If WFI or Return from ISR was used for entry: – any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*).). If WFE was used for entry and SEVONPEND = 1: – any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*) – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt ([PERIPH]EN, [PERIPH]SLPEN and [PERIPH]STPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	Wake-up on flash: – HSI as SYSCLK: HSI setup + flash setup – MSI as SYSCLK: HSI setup + flash setup + MSI setup

Stop 2 mode

The Stop 2 mode is similar to Stop 1 except that most of the core domain (D1 domain) is put in a lower leakage mode. Only the part of the core domain embedding APB3 peripherals (D2 domain) remains fully powered, allowing those peripherals to be functional.

The APB3 peripherals with the autonomous mode capability can switch on HSI16 or MSIS or MSIK for transferring data (see *Autonomous peripherals* for details).

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduced consumption.

The BOR is always available in Stop 2 mode.

I/O states in Stop 2 mode

In the Stop 2 mode, all I/O pins keep the same state as in the Run mode; except for I3C specific pull-up. PWR I3C pull-up control register 1 or 2 must be configured before entering Stop 2 mode in order to apply I3C pull-ups on corresponding I/Os. The I3C pull-up control bits must be cleared after exiting Stop 2 mode, to let I3C peripheral controls I/O pull-up activation.

Entering Stop 2 mode

The MCU enters the Stop 2 mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 85* for details on how to enter the Stop 2 mode).

If the flash memory programming is ongoing, the Stop 2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 2 mode entry is delayed until the APB access is finished.

In Stop 2 mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see *Section 9.3: PWR functional description*).
- The real-time clock (RTC) and Tamper (TAMP) kernel clock is enabled by the RTCEN bit in the RCC_BDCR.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_CSR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The I3Cx (x = 1, 2) reset pattern detection are enabled by the WUPEN9 and WUPEN10 bits in PWR_WUCR1.

Several peripherals can be autonomous in Stop 2 mode and can add consumption if they are enabled (see *Autonomous peripherals* for more details).

The COMPs, the PVM and the PVD can be used in Stop 2 mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 2), the DAC1 (two channels), the temperature sensor, the OPAMP and the VREFBUF can consume power during the Stop 2 mode, unless they are disabled before entering this mode.

Caution: All the peripherals that cannot be functional in Stop 2 mode must be either disabled by clearing the enable bit in the peripheral itself, or put under reset state by configuring the RCC registers.

Exiting Stop 2 mode

The MCU exits the Stop 2 mode as defined in *Exiting a low-power mode* (see *Table 85* for details on how to exit Stop 2 mode).

When exiting Stop 2 mode by issuing an interrupt or a wake-up event, HSI16 is selected as system clock if the bit STOPWUCK is set in RCC_CFGR1. MSIS is selected as system clock if STOPWUCK is cleared. The MSI selection allows a wake-up at higher frequency (up to 48 MHz).

Several peripherals are autonomous in Stop mode, and can generate interrupts with wake-up from Stop capability.

All peripheral clocks must be enabled to allow a wake-up from Stop interrupt (Refer to Reset and clock control (RCC) for more details).

When exiting the Stop 2 mode, the MCU is in Run mode, range 2.

Table 85. Stop 2 mode

Stop 2 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 010 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 010 in PWR_CR1
	To enter Stop 2 mode, all EXTI line pending bits and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and the program execution continues.

Table 85. Stop 2 mode (continued)

Stop 2 mode	Description
Mode exit	If WFI or Return from ISR was used for entry: – any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: – any EXTI line configured in interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (refer to *Table 128: STM32U3 series vector table*). – any EXTI line configured in event mode (refer to *Section 17.3: EXTI functional description*) – any PWR wake-up line interrupt (WUF). The interrupt source can be WKUP pins or I3C reset pattern detection. – RTC, TAMP, IWDG or any other any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt ([PERIPH]EN, [PERIPH]SLPEN and [PERIPH]STPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	Wake-up on flash: – HSI as SYSCLK: regulator setup from low-power + D1 EPOD setup + HSI setup + flash setup – MSI as SYSCLK: regulator setup from low-power + D1 EPOD setup + HSI setup + flash setup + MSI setup

Stop 3 mode

The Stop 3 mode is based on the Cortex-M33 Deepsleep mode combined with peripheral clock gating. In Stop 3 mode, all clocks in the core domain are stopped. The MSIS, MSIK, HSI16 and HSE oscillators are disabled.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduce consumption.

The BOR is always available in Stop 3 mode.

I/O states in Stop 3 mode

In the Stop 3 mode, the I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers), or with a pull-down (see PWR_PDCRx registers), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register is set. The pulldown configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same I/O. After wakeup from Stop 3 mode, the pull-up/pull-down I/O configuration remains retained based on PWR_PUCRx/PWR_PDCRx as long as APC bit is set.

Some I/Os (listed in *Section 12: General-purpose I/Os (GPIO)*) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Stop 3 mode setting their respective bit to 1 in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

I3C specific pull-ups are activated thanks to PWR I3C pull-up control registers 1 or 2. These registers must be configured before entering Stop 3 mode in order to apply I3C pull-ups on corresponding I/Os. The I3C pull-up control bits must be cleared after exiting Stop 3 mode, to let I3C peripheral controls I/O pull-up activation.

The RTC outputs on PC13 and PB2 are functional in Stop 3 mode. The 22 wake-up pins multiplexed on eight events (WKUPx, x = 1 to 8).

Entering Stop 3 mode

The MCU enters the Stop 3 mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 System Control register is set (see *Table 86* for details on how to enter the Stop 3 mode).

If the flash memory programming is ongoing, the Stop 3 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 3 mode entry is delayed until the APB access is finished.

In Stop 3 mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see *Section 9.3: PWR functional description*).
- The real-time clock (RTC) and Tamper (TAMP) kernel clock is enabled by the RTCEN bit in the RCC_BDCR.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The I3Cx (x = 1, 2) reset pattern detection are enabled by the WUPEN9 and WUPEN10 bits in PWR_WUCR1.

Exiting Stop 3 mode

The MCU exits the Stop 3 mode as described in *Exiting a low-power mode* (see *Table 86* for details on how to exit Stop 3 mode).

When exiting Stop 3 mode by issuing an interrupt or a wake-up event, HSI16 is selected as system clock if the bit STOPWUCK is set in RCC_CFGR1. MSIS is selected as system clock if STOPWUCK is cleared. The MSI selection allows a wake-up at higher frequency (up to 48 MHz).

When exiting the Stop 3 mode, the MCU is in Run mode, range 2.

When exiting Stop 3 mode, I/Os that were configured with pull-up or pull-down during Stop 3 through PWR_PUCRx or PWR_PDCRx, keep this configuration upon exiting Stop 3 mode until the APC bit in PWR_CR3 is cleared by software. Once APC is cleared, the I/Os pull-up/pull-down state is configured according to the GPIOx_PUPDR registers. The content of

the PWR_PUCRx or PWR_PDCRx registers is not lost and can be re-used for a sub-sequent entering into Stop 3 mode.

Figure 27. I/O states in Stop 3 mode

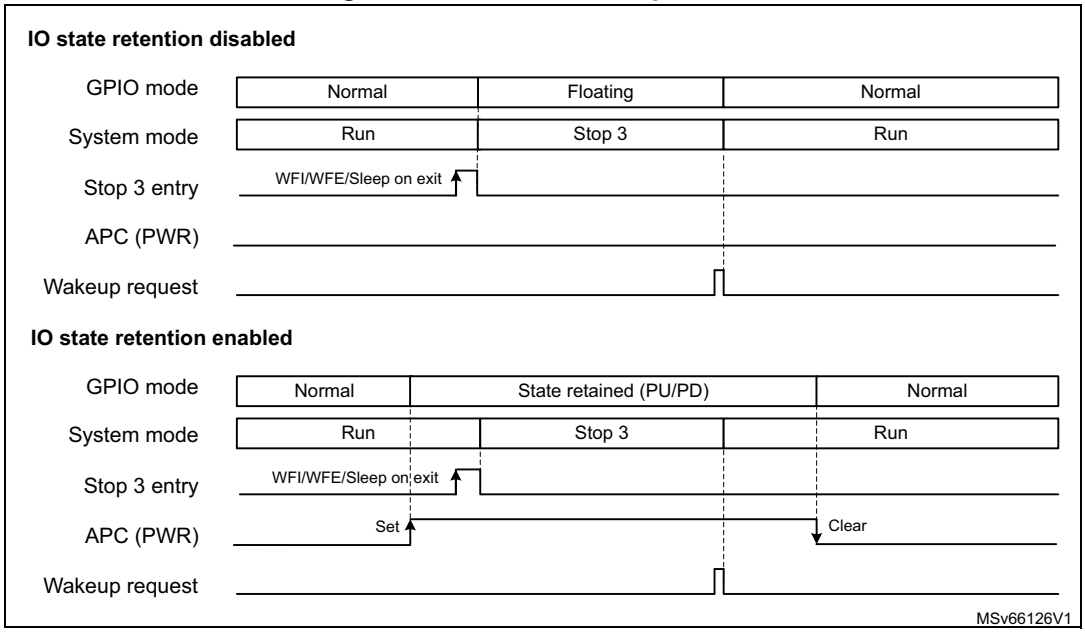


Table 86. Stop 3 mode

Stop 3 mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 011 in PWR_CR1 – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 011 in PWR_CR1 – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
	To enter Stop 3 mode, all WUFx, and the RTC/TAMP flags generating wake-up interrupts must be cleared. Otherwise, the Stop 3 mode entry procedure is completed but the Stop 3 is exited immediately after entry.

Table 86. Stop 3 mode (continued)

Stop 3 mode	Description
Mode exit	– Any PWR wake-up line interrupt (WUF). The wake-up line source can be WKUP pins edge, RTC or TAMP event, IWDG early wake-up or I3C reset pattern detection. – NRST pin external reset – BOR reset – IWDG reset
Wake-up latency	wake-up on flash: – HSI as SYSCLK: regulator setup from low-power + D1 EPOD setup + D2 EPOD setup + HSI setup + flash setup – MSI as SYSCLK: regulator setup from low-power + D1 EPOD setup + D2 EPOD setup + HSI setup + flash setup + MSI setup

Standby mode

The lowest power mode in which the BOR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, HSI16, MSIS, MSIK and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the backup domain and Standby circuitry (see *Figure 24: Power supply overview*). SRAM2 content can be partially or fully preserved depending on RRSB1, RRSB2 and RRSB3 bits configuration in PWR_CR1. In this case, the low-power regulator is ON and provides the supply to SRAM2 only.

The BOR is always available in Standby mode. The ULPMEN bit in the PWR_CR1 register must be configured to 1 to reach the lowest power consumption by forcing the BOR in ultra-low-power mode (only available when BOR level 0 is selected).

I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers), or with a pull-down (see PWR_PDCRx registers), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register is set. The pulldown configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same I/O. After wakeup from Standby mode, the pull-up/pull-down I/O configuration remains retained based on PWR_PUCRx/PWR_PDCRx as long as APC bit is set.

Some I/Os (listed in *Section 12: General-purpose I/Os (GPIO)*) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Standby mode setting their respective bit to 1 in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

I3C specific pull-ups are activated thanks to PWR I3C pull-up control registers 1 or 2. These registers must be configured before entering Standby mode in order to apply I3C pull-ups on corresponding I/Os. The I3C pull-up control bits must be cleared after exiting Standby mode, and after I3C initialization is done, to let I3C peripheral controls I/O pull-up activation.

The RTC outputs on PC13 and PB2 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. 22 wake-up pins multiplexed on eight events (WKUPx, x = 1 to 8) and the five tampers pins are available.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 87* for details on how to enter Standby mode).

In Standby mode, the following features can be selected by programming individual control bits:

. The independent watchdog (IWDG) is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset (see *Section 9.3: PWR functional description*).

- The real-time clock (RTC) and Tamper (TAMP) kernel clock is enabled by the RTCEN bit in the RCC_BDCR.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPRE bits in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The I3Cx (x = 1, 2) reset pattern detection are enabled by the WUPEN9 and WUPEN10 bits in PWR_WUCR1.

Exiting Standby mode

The MCU exits the Standby mode as described in *Exiting a low-power mode*. The SBF status flag in the PWR status register (PWR_SR) indicates that the MCU was in Standby mode.

When exiting Standby mode, I/Os that were configured with pull-up or pull-down during Standby through PWR_PUCRx or PWR_PDCRx, keep this configuration upon exiting Standby mode until the APC bit in PWR_CR3 is cleared by the software. The application can release the retained I/O state (clear the retained pull-up/pull-down) by clearing the APC bit, after reconfiguring the GPIOs and related peripherals. Once APC is cleared, the I/Os state is configured according to the GPIOx registers. The content of the PWR_PUCRx or PWR_PDCRx registers is not lost and can be re-used for a sub-sequent entering into Standby mode.

Some I/Os (listed in Section 13 General-purpose I/Os (GPIO)) are used for JTAG/SW debug and have internal pull-up or pull-down activated after reset so is configured at this reset value, as well when exiting Standby mode.

For I/Os, with a pull-up or pull-down pre-defined after reset (some JTAG/SW I/Os) or with the GPIOx_PUPDR programming done after exiting from Standby, in case those programming is different from the PWR_PUCRx or PWR_PDCRx programmed value during Standby, both a pull-down and pull-up are applied until APC is cleared, releasing the PWR_PUCRx or PWR_PDCRx programmed value.

Figure 28. I/O states in Standby mode

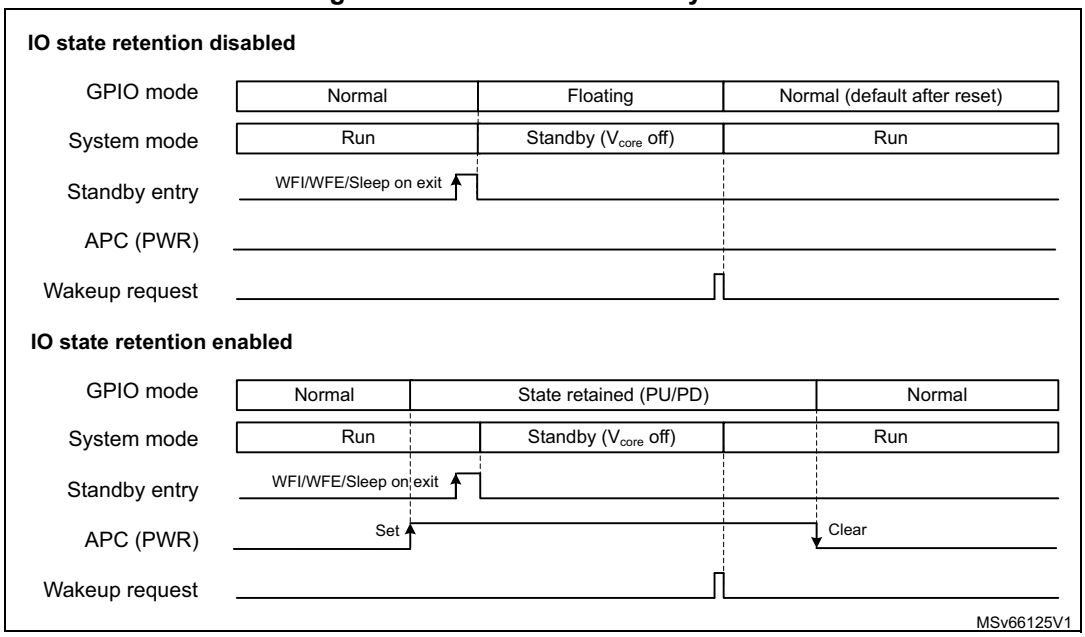


Table 87. Standby mode

Standby mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while:(1) – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 10X in PWR_CR1 – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 10X in PWR_CR1(1) – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
Mode exit	– Any PWR wake-up line interrupt (WUF). The wake-up line source can be WKUP pins edge, RTC or TAMP event, IWDG early wake-up or I3C reset pattern detection – NRST pin external reset – BOR reset – IWDG reset
Wake-up latency	Reset phase. In case SRAM2 is retained in Standby, the wake-up latency is shorter as LDO setup is from low-power, instead of from OFF state. In the case where SRAM2 is in Standby, the wake-up latency is shorter because LDO setup is done in low-power, instead of in OFF state.

1. The Standby mode is also entered if LPMS = 11X in PWR_CR1 with BREN = 1 in PWR_BDCR.

Shutdown mode

The lowest power consumption is reached in Shutdown mode. It is based on the Deepsleep mode with the voltage regulator disabled. The core domain is consequently powered off. The PLL, HSI16, MSIS, MSIK and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the backup domain. The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to backup domain is not supported.

I/O states in Shutdown mode

In the Shutdown mode, I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers (x = A, B, C, D, E, G, H)), or with a pull-down (see PWR_PDCRx registers (x = A, B, C, D, E, G, H)), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register is set. The pull-down configuration has highest priority over pull-up configuration in case both PWR_PUCRx and PWR_PDCRx are set for the same I/O. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

I3C specific pull-ups are activated thanks to PWR I3C pull-up control registers 1 or 2. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

Some I/Os (listed in *Section 12: General-purpose I/Os (GPIO)*) are used for JTAG/SW debug and can only be configured to their respective reset pull-up or pull-down state during Shutdown mode setting to 1 their respective bit in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

The RTC outputs on PC13 and PB2 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. 22 wake-up pins are multiplexed on eight events (WKUPx, x = 1 to 8) and the five RTC tampers pins are available.

Entering Shutdown mode

The MCU enters the Shutdown mode as described in *Entering a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 88* for details on how to enter Shutdown mode).

In Shutdown mode, the following features can be selected by programming individual control bits:

- The real-time clock (RTC) and Tamper (TAMP) kernel clock is enabled by the RTCEN bit in the RCC_BDCR.

Caution: In case of V_{DD} power-down, the RTC/TAMP and backup registers content is lost.

- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in the backup domain control register (RCC_BDCR).
- The I3Cx (x = 1, 2) reset pattern detection are enabled by the WUPEN9 and WUPEN10 bits in PWR_WUCR1.

Exiting Shutdown mode

The MCU exits the Shutdown mode as described in *Exiting a low-power mode*. A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the

backup domain) are reset after a wake-up from Shutdown (see *Table 88* for more details on how to exit Shutdown mode).

When exiting Shutdown mode, I/Os that were configured with pull-up or pull-down during Shutdown through registers PWR_PUCRx or PWR_PDCRx lose their configuration and are configured in floating state or to their pull-up pull-down reset value (for some I/Os listed in *Section 12: General-purpose I/Os (GPIO)*).

Table 88. Shutdown mode

Shutdown mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 11X in PWR_CR1 with BREN = 0 in PWR_BDCR – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXT = 1 – No interrupt pending – LPMS = 11X in PWR_CR1 – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
Mode exit	– Any PWR wake-up line interrupt (WUF). The wake-up line source can be WKUP pins edge, RTC or TAMP event or I3C reset pattern detection. – NRST pin external reset
Wake-up latency	Reset phase

Power mode output pins

In order to help the debug, two signals are available as device pin alternate functions:

- PWR_CSLEEP

When set, PWR_CSLEEP indicates that the CPU is in Sleep mode: WFI or WFE has been executed.

When cleared, PWR_CSLEEP indicates that the CPU is in Run mode.

- PWR_CSTOP

When set, PWR_CSTOP indicates that the device entered a Stop mode, and no autonomous peripheral requests its bus clock:

- WFI or WFE has been executed with CPU SLEEPDEEP = 1.
- No AHB/APB clock is running.

Note: The AHB/APB clocks run after WFI or WFE has been executed if an autonomous peripheral requests its bus clock in Stop mode. The peripherals bus clock request can delay or prevent the device to enter low-power modes (refer to Autonomous peripherals and Low-power modes).

The table below explains the MCU power mode depending on these signals states.

Table 89. Power modes output states versus MCU power modes

PWR_CSLEEP	PWR_CSTOP	MCU power mode(1)
0	0	Run mode
1	0	Sleep mode or Stop 0, Stop 1 or Stop 2 mode with AHB/APB clocks running
1	1	Stop 0, Stop 1 or Stop 2 mode with no AHB/APB clocks running

1. PWR_CSLEEP and PWR_CSTOP are not driven in Stop 3, Standby and Shutdown modes.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop 1, Stop 2, Stop 3, Standby or Shutdown mode while the debug features are used. This is due to the fact that the Cortex-M33 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 53.2.5: Debug and low-power modes*.

9.3.6 PWR security and privileged protection

PWR security protection

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register, some PWR register fields can be secured against nonsecure access.

The PWR TrustZone security allows the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- VBAT mode
- I/Os pull-up/pull-down configuration

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling configuration and the regulator booster (BOOSTEN) are secure.
- A GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode is secure.

Table 90 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR. As soon as at least one function is configured to be secure, the PWR clock control is also secure in the RCC.

A nonsecure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a nonsecure write access.
- The secured bits are read as 0 (RAZ) with a nonsecure read access.

A nonsecure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC. It can be read with a nonsecure read access.

When the TrustZone security is disabled (TZEN = 0), PWR_SECCFGR is RAZ/WI and all other registers are nonsecure.

Table 90. PWR Security configuration summary

Secure configuration register	Security configuration bit	Register name	Secured bits	Nonsecure access on secure bits
PWR_SECCFGR	Not applicable(1)	PWR_SECCFGR	All bits	Read OK. WI and illegal access event
PWR_SECCFGR	At least one bit is set	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_CR1	All bits	RAZ/WI
		PWR_CR2	All bits	
		PWR_SR	CSSF	WI
PWR_SECCFGR	VDMSEC	PWR_CR3	All bits	RAZ/WI
		PWR_SVMCR	All bits	RAZ/WI
PWR_SECCFGR	VBSEC	PWR_BDCR	All bits	RAZ/WI
		PWR_DBPR	All bits	RAZ/WI
PWR_SECCFGR	APCSEC	PWR_APCR	All bits	RAZ/WI
PWR_SECCFGR	WUPxSEC (x = 1 to 10)	PWR_WUCR1	WUPENx	RAZ/WI
		PWR_WUCR2	WUPPx	RAZ/WI
		PWR_WUCR3	WUSELx	RAZ/WI
		PWR_WUSCR	CWUFx	WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSR	R1EN, R2EN, BOOSTEN	RAZ/WI
GPIOx_SECCFGR (x=A,B..H)	SECy (y = 0..15)	PWR_PUCRx (x = A to H)	PUy (y = 0 to 15)	RAZ/WI
		PWR_PDCRx (x = A to H)	PDy (y = 0 to 15)	RAZ/WI
		PWR_I3CPUCR1	All bits	RAZ/WI
		PWR_I3CPUCR2	All bits	RAZ/WI

1. PWR_SECCFGR is always secure.

PWR privileged protection

By default, after a reset, all PWR registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR that can be written with privileged access only. PWR_PRIVCFGR can be read by secure and nonsecure, privileged and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC or GPIO as shown in *Table 90: PWR Security configuration summary*).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written with privileged access only, secure or nonsecure. This bit configures the privileged access of all PWR securable functions that are configured as nonsecure (defined by PWR_SECCFGR, GTZC, RCC or GPIO as shown in *Table 90: PWR Security configuration summary*).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits that are configured as nonsecure, can be written only with privileged access.
- The PWR securable bits that are configured as nonsecure, can be read only with privileged access except PWR_PRIVCFGR that can be read by privileged or unprivileged accesses.
- The R1RDY, R2RDY and BOOSTRDY bits in PWR_VOSR, PWR_SR, PWR_SVMSR and PWR_WUSR, can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

9.4 PWR interrupts

The table below gives a summary of the interrupt sources and the way to control them.

Table 91. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop 0, 1, 2	Exit Stop 3, Standby, Shutdown
PWR(1)	Nonsecure wake-up interrupt	WUFx and (WUSELx ≠ 11) (x = 1 to 10)	WUPENx and (WUPxSEC = 0) (x = 1 to 10)	Write CWUFx = 1 (x = 1 to 10)	Yes	Yes
PWR_S(1)	Secure wake-up interrupt	WUFx and (WUSELx ≠ 11) (x = 1 to 10)	WUPENx and (WUPxSEC = 1) (x = 1 to 10)	Write CWUFx = 1 (x = 1 to 10)	Yes	Yes

Table 91. PWR interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop 0, 1, 2	Exit Stop 3, Standby, Shutdown
PVD_PVM	Programmable voltage detector through EXTI line16	PVDO	EXTI line 16 enabled	Write EXTI PIF16 = 1	Yes	No
	USB supply voltage monitor through EXTI line 19	VDDUSBRDY	EXTI line 19 enabled	Write EXTI PIF19 = 1	Yes	No
	VDDIO2 supply voltage monitor through EXTI line20	VDDIO2RDY	EXTI line 20 enabled	Write EXTI PIF20 = 1		
	Analog supply voltage monitor1 through EXTI line21	VDDA1RDY	EXTI line 21 enabled	Write EXTI PIF21 = 1		
	Analog supply voltage monitor2 through EXTI line22	VDDA2RDY	EXTI line 22 enabled	Write EXTI PIF22 = 1		

1. This interrupt is not generated for RTC, TAMP and IWDG internal wake-up lines.

9.5 PWR registers

The PWR registers can be accessed in word, half-word, and byte format, unless otherwise specified.

9.5.1 PWR control register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Power-on reset value: 0x0000 0U00

This register is not affected by Standby mode.

This register is protected against nonsecure access when LPMSEC = 1 in PWR_SECCFGR. This register is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SRAM2PD	SRAM1PD	ULPMEN	RRSB3	RRSB2	RRSB1	Res.	LPMS[2:0]		
						rs	rs	rw	rw	rw	rw		rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **SRAM2PD**: SRAM2 power down

This bit is used to reduce the consumption by definitively powering off the SRAM2 until the next power-on reset.

0: SRAM2 powered on

1: SRAM2 powered off

Note: This bit is cleared upon power-on reset.

Bit 8 **SRAM1PD**: SRAM1 power down

This bit is used to reduce the consumption by definitively powering off the SRAM1 until the next power-on reset.

0: SRAM1 powered on

1: SRAM1 powered off

Note: This bit is cleared upon power-on reset.

Bit 7 **ULPMEN**: BOR0 ultra-low power mode

This bit is used to reduce the consumption by configuring the BOR in discontinuous mode. This bit has effect only when the BOR level 0 is selected, and when the device is in Standby mode. This bit must be set to reach the lowest power consumption in Standby mode.

0: BOR level 0 operating in continuous (normal) mode in Standby mode

1: BOR level 0 operating in discontinuous (ultra-low power) mode in Standby mode

Bit 6 **RRSB3**: SRAM2 page 3 retention in Standby mode

This bit is used to keep the SRAM2 page 3 content in Standby mode. The SRAM2 page 3 corresponds to the last 8 Kbytes of the SRAM2 (from SRAM2 base address + 0xE000 to SRAM2 base address + 0xFFFF).

0: SRAM2 page3 content not retained in Standby mode

1: SRAM2 page3 content retained in Standby mode

Note: This bit has no effect in Shutdown mode.

Bit 5 **RRSB2**: SRAM2 page 2 retention in Standby mode

This bit is used to keep the SRAM2 page 2 content in Standby mode. The SRAM2 page 2 corresponds to the 24 Kbytes of the SRAM2 (from SRAM2 base address + 0x8000 to SRAM2 base address + 0xDFFF).

0: SRAM2 page2 content not retained in Standby mode

1: SRAM2 page2 content retained in Standby mode

Note: This bit has no effect in Shutdown mode.

Bit 4 **RRSB1**: SRAM2 page 1 retention in Standby mode

This bit is used to keep the SRAM2 page 1 content in Standby mode. The SRAM2 page 1 corresponds to the first 32 Kbytes of the SRAM2 (from SRAM2 base address to SRAM2 base address + 0x7FFF).

0: SRAM2 page1 content not retained in Standby mode

1: SRAM2 page1 content retained in Standby mode

Note: This bit has no effect in Shutdown mode.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits select the low-power mode entered when the CPU enters Deepsleep mode.

000: Stop 0 mode

001: Stop 1 mode

010: Stop 2 mode

011: Stop 3 mode

100-101: Standby mode

110-111: Shutdown mode

9.5.2 PWR control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

This register is protected against nonsecure access when LPMSEC = 1 in PWR_SECCFGR. This register is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FLASHFWU	SRAMFWU	PKARAMPDS	PRAMPDS	ICRAMPDS	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2PDS3	SRAM2PDS2	SRAM2PDS1
		rw	rw	rw	rw	rw							rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1PDS7	SRAM1PDS6	SRAM1PDS5	SRAM1PDS4	SRAM1PDS3	SRAM1PDS2	SRAM1PDS1
									rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FLASHFWU**: Flash memory fast wake-up from Stop 0 and Stop 1 modes

This bit is used to obtain the best trade-off between low-power consumption and wake-up time when exiting Stop 0 or Stop 1 modes. When this bit is set, the flash memory remains in normal mode in Stop 0 and Stop 1 modes, which offers a faster startup time with higher consumption.

0: Flash memory enters low-power mode in Stop 0 and Stop 1 modes (lower-power consumption).

1: Flash memory remains in normal mode in Stop 0 and Stop 1 modes (faster wake-up time).

Bit 28 **SRAMFWU**: SRAM fast wake-up from Stop 0 and Stop 1 modes

This bit is used to obtain the best trade-off between low-power consumption and wake-up time. SRAMs wake-up time increases the wake-up time when exiting Stop 0 and 1 modes, and also increases the GPDMA1 access time to SRAMs during Stop modes.

0: SRAMs enter low-power mode in Stop 0 and Stop 1 modes (source biasing for lower-power consumption).

1: SRAMs remain in normal mode in Stop 0 and Stop 1 modes (higher consumption but no SRAM wake-up time).

Note: If one or several SRAMs are configured to be in power-down in Stop 1, Stop 2, and Stop 3 modes through the xRAMyPDS bits, setting this bit has no effect in Stop 1 mode.

Bit 27 **PKARAMPDS**: PKA SRAM power-down in Stop 1, Stop 2, and Stop 3 modes

0: PKA SRAM content retained in Stop modes

1: PKA SRAM content lost in Stop modes

Bit 26 **PRAMPDS**: FDCAN and USB SRAM power-down in Stop 1, Stop 2, and Stop 3 modes

0: FDCAN and USB SRAM content retained in Stop modes

1: FDCAN and USB SRAM content lost in Stop modes

Bit 25 **ICRAMPDS**: ICACHE SRAM power-down in Stop 1, Stop 2, and Stop 3 modes

0: ICACHE SRAM content retained in Stop modes

1: ICACHE SRAM content lost in Stop modes

Bits 24:19 Reserved, must be kept at reset value.

Bit 18 **SRAM2PDS3**: SRAM2 page 3 (8 Kbytes) power-down in Stop 1, Stop 2, and Stop 3 modes
0: SRAM2 page 3 content retained in Stop modes
1: SRAM2 page 3 content lost in Stop modes

Bit 17 **SRAM2PDS2**: SRAM2 page 2 (24 Kbytes) power-down in Stop 1, Stop 2, and Stop 3 modes
0: SRAM2 page 2 content retained in Stop modes
1: SRAM2 page 2 content lost in Stop modes

Bit 16 **SRAM2PDS1**: SRAM2 page 1 (32 Kbytes) power-down in Stop 1, Stop 2, and Stop 3 modes
0: SRAM2 page 1 content retained in Stop modes
1: SRAM2 page 1 content lost in Stop modes

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **SRAM1PDSi**: SRAM1 page i power-down in Stop 1, Stop 2, and Stop 3 modes (i = 7 to 1)
0: SRAM1 page i content retained in Stop modes
1: SRAM1 page i content lost in Stop modes
Note: Page 1 to 2 size is 16 Kbytes. Page 3 to 7 size is 32 Kbytes.

9.5.3 PWR control register 3 (PWR_CR3)

Address offset: 0x08

System reset value: 0b0000 0000 0000 0000 0000 0000 0000 0U00

Power-on reset value: 0x0000 0000

This register is not affected by Standby mode.

This register is protected against nonsecure access when VDMSEC = 1 in PWR_SECCFGR. This register is protected against unprivileged access when VDMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VDMSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTEN	REGSEL	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSTEN**: Fast soft start
0: LDO/SMPS fast startup disabled (limited inrush current)
1: LDO/SMPS fast startup enabled

Bit 1 **REGSEL**: Regulator selection
0: LDO selected
1: SMPS selected
Note: REGSEL is reserved and must be kept at reset value in packages without SMPS.

Bit 0 Reserved, must be kept at reset value.

9.5.4 PWR voltage scaling register (PWR_VOSR)

Address offset: 0x0C

Reset value: 0x0002 0002

Some register bits are protected against nonsecure access depending on RCC_SECCFGR.

These bits can be protected against unprivileged access depending on PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTRDY	Res.	Res.	Res.	Res.	Res.	Res.	R2RDY	R1RDY
							r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN	Res.	Res.	Res.	Res.	Res.	Res.	R2EN	R1EN
							rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **BOOSTRDY**: EPOD booster ready

This bit is set to 1 by hardware when the power booster startup time is reached. The system clock frequency can be switched higher than 24 MHz only after this bit is set. Disabling the booster clock when the booster is ready is forbidden.

0: Power booster not ready

1: Power booster ready

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **R2RDY**: Ready bit for V_{CORE} voltage range 2 selection

0: Range 2 not ready: voltage level < VOS range 2 level

1: Range 2 ready: voltage level ≥ VOS range 2 level

Note: R1RDY and R2RDY cannot be set at the same time.

Bit 16 **R1RDY**: Ready bit for V_{CORE} voltage range 1 selection

0: Range 1 not ready: voltage level < VOS range 1 level

1: Range 1 ready: voltage level ≥ VOS range 1 level

Note: R1RDY and R2RDY cannot be set at the same time

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN**: EPOD booster enable

This bit is protected against nonsecure access when SYSCLKSEC = 1 in RCC_SECCFGR. It is protected against unprivileged access when SYSCLKSEC = 1 in RCC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SYSCLKSEC = 0 and NSPRIV = 1.

This bit must be set in range 1, and before increasing the system clock frequency above 24 MHz in range 2. The booster clock must be configured before setting this bit, and must not be disabled as long as the booster is enabled.

0: Booster disabled

1: Booster enabled

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **R2EN**: Voltage scaling range 2 selection

This bit is protected against nonsecure access when SYSCLKSEC = 1 in RCC_SECCFGR. It is protected against unprivileged access when SYSCLKSEC = 1 in RCC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SYSCLKSEC = 0 and NSPRIV = 1.

0: Voltage scaling range 2 disabled

1: Voltage scaling range 2 enabled

Note: R1EN and R2EN must be at opposite value. Any attempt to write R1EN and R2EN to the same value is ignored. Modifying R1EN and R2EN is possible only when current range is ready (R1RDY = R1EN and R2RDY = R2EN).

Bit 0 **R1EN**: Voltage scaling range 1 selection

This bit is protected against nonsecure access when SYSCLKSEC = 1 in RCC_SECCFGR. It is protected against unprivileged access when SYSCLKSEC = 1 in RCC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SYSCLKSEC = 0 and NSPRIV = 1.

0: Voltage scaling range 1 disabled

1: Voltage scaling range 1 enabled

Note: R1EN and R2EN must be at opposite value. Any attempt to write R1EN and R2EN to the same value is ignored. Modifying R1EN and R2EN is possible only when current range is ready (R1RDY = R1EN and R2RDY = R2EN).

9.5.5 PWR supply voltage monitoring control register (PWR_SVMCR)

Address offset: 0x10

Reset value: 0x0000 0000

This register is protected against nonsecure access when VDMSEC = 1 in PWR_SECCFGR. This register is protected against unprivileged access when VDMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VDMSEC = 0 and NSPRIV = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ASV	IO2SV	USV	AVM2EN	AVM1EN	IO2VMEN	UVMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS[2:0]			PVDE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ASV**: V_{DDA} independent analog supply valid

This bit is used to validate the V_{DDA} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the analog peripherals. If V_{DDA} is not always present in the application, the V_{DDA} voltage monitor can be used to determine whether this supply is ready or not.

0: V_{DDA} not present: logical and electrical isolation is applied to ignore this supply.

1: V_{DDA} valid

Bit 29 **IO2SV**: V_{DDIO2} independent I/O supply valid

This bit is used to validate the V_{DDIO2} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PG[15:2]. If V_{DDIO2} is not always present in the application, the V_{DDIO2} voltage monitor can be used to determine whether this supply is ready or not.

0: V_{DDIO2} not present: logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO2} valid

Bit 28 **USV**: V_{DDUSB} independent USB supply valid
This bit is used to validate the V_{DDUSB} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB peripheral. If V_{DDUSB} is not always present in the application, the V_{DDUSB} voltage monitor can be used to determine whether this supply is ready or not.
0: V_{DDUSB} not present: logical and electrical isolation is applied to ignore this supply.
1: V_{DDUSB} valid

Bit 27 **AVM2EN**: V_{DDA} independent analog supply voltage monitor 2 enable (1.8 V threshold)
0: V_{DDA} voltage monitor 2 disabled
1: V_{DDA} voltage monitor 2 enabled

Bit 26 **AVM1EN**: V_{DDA} independent analog supply voltage monitor 1 enable (1.6 V threshold)
0: V_{DDA} voltage monitor 1 disabled
1: V_{DDA} voltage monitor 1 enabled

Bit 25 **IO2VMEN**: V_{DDIO2} independent I/O voltage monitor enable
0: V_{DDIO2} voltage monitor disabled
1: V_{DDIO2} voltage monitor enabled

Bit 24 **UVMEN**: V_{DDUSB}independent USB voltage monitor enable
0: V_{DDUSB} voltage monitor disabled
1: V_{DDUSB} voltage monitor enabled

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:5 **PVDLS[2:0]**: Programmable voltage detector (PVD) level selection
These bits select the voltage threshold detected by the PVD.
000: V_{PVD0} around 2.0 V
001: V_{PVD1} around 2.2 V
010: V_{PVD2} around 2.4 V
011: V_{PVD3} around 2.5 V
100: V_{PVD4} around 2.6 V
101: V_{PVD5} around 2.8 V
110: V_{PVD6} around 2.9 V
111: External input analog voltage PVD_IN (compared internally to VREFINT)

Bit 4 **PVDE**: Programmable voltage detector enable
0: PVD disabled
1: PVD enabled

Bits 3:0 Reserved, must be kept at reset value.

9.5.6 PWR wake-up control register 1 (PWR_WUCR1)

Address offset: 0x14

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit WUPENi is protected against nonsecure access when WUPiSEC = 1 in PWR_SECCFGR.

Each bit WUPENi is protected against unprivileged access when WUPiSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPiSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	WUPEN10	WUPEN9	WUPEN8	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **WUPENi**: Wake-up line WKUPi enable (i = 10 to 1)
0: Wake-up line WKUPi disabled
1: Wake-up line WKUPi enabled

9.5.7 PWR wake-up control register 2 (PWR_WUCR2)

Address offset: 0x18

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit WUPPi is protected against nonsecure access when WUPiSEC = 1 in PWR_SECCFGR. Each bit WUPPi is protected against unprivileged access when WUPiSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPiSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPP8	WUPP7	WUPP6	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **WUPPi**: Wake-up line WKUP8 polarity (i = 8 to 1)
This bit must be configured when WUPENi = 0. It has no effect when WUSELi[1:0] = 11.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

9.5.8 PWR wake-up control register 3 (PWR_WUCR3)

Address offset: 0x1C

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bitfield WUSELi[1:0] is protected against nonsecure access when WUPiSEC = 1 in PWR_SECCFGR. Each bitfield WUSELi[1:0] is protected against unprivileged access when WUPiSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPiSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUSEL8[1:0]		WUSEL7[1:0]		WUSEL6[1:0]		WUSEL5[1:0]		WUSEL4[1:0]		WUSEL3[1:0]		WUSEL2[1:0]		WUSEL1[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WUSELi[1:0]**: Wake-up line WKUPi selection (i = 8 to 1)
This bitfield must be configured when WUPENi = 0.
00: WKUPi_0
01: WKUPi_1
10: WKUPi_2
11: WKUPi_3

9.5.9 PWR backup domain control register (PWR_BDCR)

Address offset: 0x24

System reset value: 0xUUUU UUUU

Backup domain reset value: 0x0000 0000

This register is not affected by Standby mode.

This register is protected against nonsecure access when VBSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VBSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBRS**: V_{BAT} charging resistor selection
0: V_{BAT} charged through a 5 kΩ resistor
1: V_{BAT} charged through a 1.5 kΩ resistor

Bit 0 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled
1: V_{BAT} battery charging enabled

9.5.10 PWR disable backup domain register (PWR_DBPR)

Address offset: 0x28

Reset value: 0x0000 0000

This register is protected against nonsecure access when VBSEC = 1 in PWR_SECCFGR.
This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VBSEC = 0 and NSPRIV = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DBP**: Disable backup domain write protection

In reset state, all registers and SRAM in backup domain are protected against parasitic write access. This bit must be set to enable the write access to these registers.

0: Write access to backup domain disabled

1: Write access to backup domain enabled

9.5.11 PWR security configuration register (PWR_SECCFGR)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be written only when the access is secure. It can be read by secure or nonsecure access. This register is write-protected against unprivileged write access when SPRIV = 1 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APC SEC	VB SEC	VDM SEC	LPM SEC	Res.	Res.	WUP10 SEC	WUP9 SEC	WUP8 SEC	WUP7 SEC	WUP6 SEC	WUP5 SEC	WUP4 SEC	WUP3 SEC	WUP2 SEC	WUP1 SEC
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **APCSEC**: Pull-up/pull-down secure protection

0: PWR_APCR can be read and written with secure or nonsecure access.

1: PWR_APCR can be read and written only with secure access.

Bit 14 **VBSEC**: Backup domain secure protection

0: PWR_BDCR and PWR_DBPR can be read and written with secure or nonsecure access.

1: PWR_BDCR and PWR_DBPR can be read and written only with secure access.

Bit 13 **VDMSEC**: Voltage detection and monitoring secure protection

0: PWR_SVMCR and PWR_CR3 can be read and written with secure or nonsecure access.

1: PWR_SVMCR and PWR_CR3 can be read and written only with secure access.

Bit 12 **LPMSEC**: Low-power mode secure protection

0: PWR_CR1, PWR_CR2, and CSSF bit in the PWR_SR can be read and written with secure or nonsecure access.

1: PWR_CR1, PWR_CR2, and CSSF bit in the PWR_SR can be read and written only with secure access.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **WUPiSEC**: WUPi secure protection (i = 10 to 1)

0: Bits related to the WKUPi line in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3, and PWR_WUSCR can be read and written with secure or nonsecure access.

1: Bits related to the WKUPi line in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3, and PWR_WUSCR can be read and written only with secure access.

9.5.12 PWR privilege control register (PWR_PRIVCFGR)

Address offset: 0x34

Reset value: 0x0000 0000

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NS PRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: PWR nonsecure function privilege configuration

This bit is set and reset by software. It can be written only by privileged access, secure or nonsecure.

0: Read and write to PWR nonsecure functions can be done by privileged or unprivileged access.

1: Read and write to PWR nonsecure functions can be done by privileged access only.

Bit 0 **SPRIV**: PWR secure function privilege configuration

This bit is set and reset by software. It can be written only by a secure privileged access.

0: Read and write to PWR secure functions can be done by privileged or unprivileged access.

1: Read and write to PWR secure functions can be done by privileged access only.

9.5.13 PWR status register (PWR_SR)

Address offset: 0x38

Reset value: 0x0000 0000

Some bits are protected against nonsecure access depending on PWR_SECCFGR. Some bits are protected against unprivileged access depending on PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	CSSF
													r	r	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SBF**: Standby flag

This bit is set by hardware when the device enters the Standby mode, and is cleared by writing 1 to the CSSF bit, or by a power-on reset. It is not cleared by the system reset.

0: The device did not enter Standby mode.

1: The device entered Standby mode.

Bit 1 **STOPF**: Stop flag

This bit is set by hardware when the device enters a Stop mode, and is cleared by software by writing 1 to the CSSF bit.

0: The device did not enter any Stop mode.

1: The device entered a Stop mode.

Bit 0 **CSSF**: Clear Stop and Standby flags

This bit is protected against nonsecure access when LPMSEC = 1 in PWR_SECCFGR.
This bit is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.
Writing 1 to this bit clears the STOPF and SBF flags.

9.5.14 PWR supply voltage monitoring status register (PWR_SVMSR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VDDA2RDY	VDDA1RDY	VDDIO2RDY	VDDUSBRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
											r			r	

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **VDDA2RDY**: V_{DDA} ready versus 1.8 V voltage monitor

0: V_{DDA} is below the threshold of the V_{DDA} voltage monitor 2 (around 1.8 V).

1: V_{DDA} is equal or above the threshold of the V_{DDA} voltage monitor 2 (around 1.8 V).

Bit 26 **VDDA1RDY**: V_{DDA} ready versus 1.6 V voltage monitor

0: V_{DDA} is below the threshold of the V_{DDA} voltage monitor 1 (around 1.6 V).

1: V_{DDA} is equal or above the threshold of the V_{DDA} voltage monitor 1 (around 1.6 V).

Bit 25 **VDDIO2RDY**: V_{DDIO2} ready

0: V_{DDIO2} is below the threshold of the V_{DDIO2} voltage monitor.

1: V_{DDIO2} is equal or above the threshold of the V_{DDIO2} voltage monitor.

Bit 24 **VDDUSBRDY**: V_{DDUSB} ready

0: V_{DDUSB} is below the threshold of the V_{DDUSB} voltage monitor.

1: V_{DDUSB} is equal or above the threshold of the V_{DDUSB} voltage monitor.

Bits 23:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable voltage detector output

0: V_{DD} is equal or above the PVD threshold selected by PVDLS[2:0].

1: V_{DD} is below the PVD threshold selected by PVDLS[2:0].

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **REGS**: Regulator selection

0: LDO selected

1: SMPS selected

Bit 0 Reserved, must be kept at reset value.

9.5.15 PWR wake-up status register (PWR_WUSR)

Address offset: 0x44

Reset value: 0x0000 0000

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **WUFi**: Wake-up flag i (i = 10 to 1)

This bit is set when a wake-up event is detected on WKUPi line. This bit is cleared by writing 1 in the CWUFi bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPENi = 0. If WUSEL = 11, this bit is cleared by hardware when all internal wake-up source are cleared.

9.5.16 PWR wake-up status clear register (PWR_WUSCR)

Address offset: 0x48

Reset value: 0x0000 0000

Each bit CWUFi is protected against nonsecure access when WUPiSEC = 1 in PWR_SECCFGR. Each bit CWUFi is protected against unprivileged access when WUPiSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPiSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CWUF 10	CWUF 9	CWUF 8	CWUF 7	CWUF 6	CWUF 5	CWUF 4	CWUF 3	CWUF 2	CWUF 1
						w	w	w	w	w	w	w	w	w	w

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **CWUFi**: Wake-up flag i clear (i = 10 to 1)

Writing 1 to this bit clears the WUFi flag in PWR_WUSR.

9.5.17 PWR apply pull configuration register (PWR_APCR)

Address offset: 0x4C

Reset value: 0x0000 0000

This register is not affected by Standby mode.

This register is protected against nonsecure access when APCSEC = 1 in PWR_SECCFGR. This register is protected against unprivileged access when APCSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when APCSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **APC**: Apply pull-up and pull-down configuration

When this bit is set, the I/O pull-up and pull-down configurations defined in PWR_PUCRx and PWR_PDCRx are applied.

When this bit is cleared, PWR_PUCRx and PWR_PDCRx are not applied to the I/Os.

Note: When APC is set, I/Os configurations from GPIO registers are still applied in Run and Sleep modes (ORed with PWR registers). Care must be taken to define a coherent I/O configuration in GPIO and PWR pull-up/pull-down control registers.

9.5.18 PWR port A pull-up control register (PWR_PUCRA)

Address offset: 0x50

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PUy is protected against nonsecure access when SECy = 1 in GPIOA_SECCFGR. Each bit PUy is protected against unprivileged access when SECy = 1 in GPIOA_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	Res.	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PU15**: Port A pull-up bit 15

When set, each bit activates the pull-up on PA15 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PD15 bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bit 14 Reserved, must be kept at reset value.

Bits 13:0 **PUy**: Port A pull-up bit y (y = 13 to 0)

When set, each bit activates the pull-up on PAy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.19 PWR port A pull-down control register (PWR_PDCRA)

Address offset: 0x54

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PDy is protected against nonsecure access when SECy = 1 in GPIOA_SECCFGR.
Each bit PDy is protected against unprivileged access when SECy = 1 in GPIOA_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PD14	Res.	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **PD14**: Port A pull-down bit 14

When set, each bit activates the pull-down on PA14 when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **PDy**: Port A pull-down bit y (y = 12 to 0)

When set, each bit activates the pull-down on PAy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.20 PWR port B pull-down control register (PWR_PDCRB)

Address offset: 0x5C

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PDy is protected against nonsecure access when SECy = 1 in GPIOB_SECCFGR.
Each bit PDy is protected against unprivileged access when SECy = 1
in GPIOB_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	Res.	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **PDy**: Port B pull-down bit y (y = 15 to 5)

When set, each bit activates the pull-down on PBy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **PDy**: Port B pull-down bit y (y = 3 to 0)

When set, each bit activates the pull-down on PBy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.21 PWR port x pull-up control register (PWR_PUCRx) (x = B to E)

Address offset: Port B: 0x58

Address offset: Port C: 0x60

Address offset: Port D: 0x68

Address offset: Port E: 0x70

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PUy is protected against nonsecure access when SECy = 1 in GPIOx_SECCFGR.

Each bit PUy is protected against unprivileged access when SECy = 1 in GPIOx_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SEC y = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port x pull-up bit y (y = 15 to 0)

When set, each bit activates the pull-up on Pxy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.22 PWR port x pull-down control register (PWR_PDCRx) (x = C to E)

Address offset: Port C: 0x64

Address offset: Port D: 0x6C

Address offset: Port E: 0x74

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PDy is protected against nonsecure access when SECy = 1 in GPIOx_SECCFGR.

Each bit PDy is protected against unprivileged access when SECy = 1 in GPIOx_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Note:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port x pull-down bit y (y = 15 to 0)

When set, each bit activates the pull-down on Pxy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.23 PWR port G pull-up control register (PWR_PUCRG)

Address offset: 0x80

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PUy is protected against nonsecure access when SECy = 1 in GPIOG_SECCFGR.

Each bit PUy is protected against unprivileged access when SECy = 1 in GPIOG_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **PUy**: Port G pull-up bit y (y = 15 to 2)

When set, each bit activates the pull-up on PGy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 1:0 Reserved, must be kept at reset value.

9.5.24 PWR port G pull-down control register (PWR_PDCRG)

Address offset: 0x84

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PDy is protected against nonsecure access when SECy = 1 in GPIOG_SECCFGR.

Each bit PDy is protected against unprivileged access when SECy = 1 in GPIOG_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **PDy**: Port G pull-down bit y (y = 15 to 2)

When set, each bit activates the pull-down on PGy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 1:0 Reserved, must be kept at reset value.

9.5.25 PWR port H pull-up control register (PWR_PUCRH)

Address offset: 0x88

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PUy is protected against nonsecure access when SECy = 1 in GPIOH_SECCFGR.

Each bit PUy is protected against unprivileged access when SECy = 1 in GPIOH_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU3	Res.	PU1	PU0
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU3**: Port H pull-up bit 3

When set, each bit activates the pull-up on PH3 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PD3 bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PUy**: Port H pull-up bit y (y = 1 to 0)

When set, each bit activates the pull-up on PHy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.26 PWR port H pull-down control register (PWR_PDCRH)

Address offset: 0x8C

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit PDy is protected against nonsecure access when SECy = 1 in GPIOH_SECCFGR.
Each bit PDy is protected against unprivileged access when SECy = 1

in GPIOH_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD3	Res.	PD1	PD0
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PD3**: Port H pull-down bit 3

When set, each bit activates the pull-down on PH3 when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PDy**: Port H pull-down bit y (y = 1 to 0)

When set, each bit activates the pull-down on PHy when the APC bit is set in PWR_APCR.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

9.5.27 PWR I3C pull-up control register 1 (PWR_I3CPUCR1)

Address offset: 0xB0

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit Pxy_I3CPU is protected against nonsecure access when SECy = 1 in GPIOx_SECCFGR.

Each bit Pxy_I3CPU is protected against unprivileged access when SECy = 1 in GPIOx_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PB14_I3CPU	PB13_I3CPU	PB12_I3CPU	PB10_I3CPU	PB9_I3CPU	PB8_I3CPU	PB6_I3CPU	PB2_I3CPU	Res.	PA7_I3CPU	PA6_I3CPU	PA1_I3CPU
				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:9 **PBx_I3CPU**: Port B pin x I3C pull-up (x = 14 to 12)

When set, the bit activates the I3C pull-up on PBx.

Bits 8:6 **PBy_I3CPU**: Port B pin y I3C pull-up (y = 10 to 8)
When set, the bit activates the I3C pull-up on PBy.

Bit 5 **PB6_I3CPU**: Port B pin 6 I3C pull-up
When set, the bit activates the I3C pull-up on PB6.

Bit 4 **PB2_I3CPU**: Port B pin 2 I3C pull-up
When set, the bit activates the I3C pull-up on PB2.

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **PAy_I3CPU**: Port A pin y I3C pull-up (y = 7 to 6)
When set, the bit activates the I3C pull-up on PAy.

Bit 0 **PA1_I3CPU**: Port A pin 1 I3C pull-up
When set, the bit activates the I3C pull-up on PA1.

9.5.28 PWR I3C pull-up control register 2 (PWR_I3CPUCR2)

Address offset: 0xB4

Reset value: 0x0000 0000

This register is not affected by Standby mode.

Each bit Pxy_I3CPU is protected against nonsecure access when SECy = 1 in GPIOx_SECCFGR.

Each bit Pxy_I3CPU is protected against unprivileged access when SECy = 1 in GPIOx_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PH3_I3CPU	Res.	PG14_I3CPU	PG13_I3CPU	PG8_I3CPU	PG7_I3CPU	Res.	PD13_I3CPU	PD12_I3CPU	Res.	PC1_I3CPU	PC0_I3CPU
				rw		rw	rw	rw	rw		rw	rw		rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PH3_I3CPU**: Port H pin 3 I3C pull-up
When set, the bit activates the I3C pull-up on PH3.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **PGx_I3CPU**: Port G pin x I3C pull-up (x = 14 to 13)
When set, the bit activates the I3C pull-up on PGx.

Bits 7:6 **PGy_I3CPU**: Port G pin y I3C pull-up (y = 8 to 7)
When set, the bit activates the I3C pull-up on PGy.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **PDy_I3CPU**: Port D pin y I3C pull-up (y = 13 to 12)
When set, the bit activates the I3C pull-up on PDy.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PCy_I3CPU**: Port C pin 1 I3C pull-up (y = 1 to 0)
When set, the bit activates the I3C pull-up on PCy.

9.5.29 PWR register map

Table 92. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2PD	SRAM1PD	ULPMEN	RRSB3	RRSB2	RRSB1	Res.	LPMS[2:0]		
	Reset value																							0	0	0	0	0	0		0	0	0
0x04	PWR_CR2	Res.	Res.	FLASHFWU	SRAMFWU	PKARAMPDS	PRAMPDS	ICRAMPDS	Res.	Res.	Res.	Res.	Res.	Res.	SRAM2PDS3	SRAM2PDS2	SRAM2PDS1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM1PDS7	SRAM1PDS6	SRAM1PDS5	SRAM1PDS4	SRAM1PDS3	SRAM1PDS2	SRAM1PDS1
	Reset value			0	0	0	0	0							0	0	0									0	0	0	0	0	0	0
0x08	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTEN	REGSEL	Res.
	Reset value																														0	0	
0x0C	PWR_VOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTRDY	Res.	Res.	Res.	Res.	Res.	Res.	R2RDY	R1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN					R2EN	R1EN	
	Reset value								0							1	0									0						1	0
0x10	PWR_SVMCR	Res.	ASV	IO2SV	USV	AVM2EN	AVM1EN	IO2VMEN	UVMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS[2:0]			PDVE	Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0																	0	0	0	0				
0x14	PWR_WUCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN10	WUPEN9	WUPEN8	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x18	PWR_WUCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUSEL8 [1:0]		WUSEL7 [1:0]		WUSEL6 [1:0]		WUSEL5 [1:0]		WUSEL4 [1:0]		WUSEL3 [1:0]		WUSEL2 [1:0]		WUSEL1 [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	PWR_WUCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUP8	WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1
	Reset value																									0	0	0	0	0	0	0	0
0x20	Reserved	Res.																															
0x24	PWR_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE
	Reset value																															0	0
0x28	PWR_DBPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x2C	Reserved	Res.																															

Table 92. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APCSEC	VBSEC	VDMSEC	LPMSEC	Res.	Res.	WUP10SEC	WUP9SEC	WUP8SEC	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
	Reset value																	0	0	0	0			0	0	0	0	0	0	0	0	0	0
0x34	PWR_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0
0x38	PWR_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	CSSF
	Reset value																														0	0	0
0x3C	PWR_SVMR	Res.	Res.	Res.	Res.	VDDA2RDY	VDDA1RDY	VDDIO2RDY	VDDUSBRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
	Reset value					0	0	0	0																				0			0	
0x30-0x40	Reserved	Res.																															
0x44	PWR_WUSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF10	WUF9	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	PWR_WUSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF10	CWUF9	CWUF8	CWUF7	CWUF6	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x4C	PWR_APCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC
	Reset value																																0
0x50	PWR_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	Res.	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD14	Res.	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																		0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x58 +0x8 * z (z=0 to 3)	PWR_PUCRx (x = B to E)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	PWR_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	Res.	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0
0x64 +0x8 * z (z=0 to 2)	PWR_PDCRx (x = C to E)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	PWR_PUCRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x84	PWR_PDCRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x88	PWR_PUCRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU3	Res.	PU1	PU0
	Reset value																													0		0	0

Table 92. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8C	PWR_PDCRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD3	Res.	PD1	PD0
	Reset value																													0		0	0
0xA0-0xAC	Reserved	Res.																															
0xB0	PWR_I3CPUCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB14_I3CPU	PB13_I3CPU	PB12_I3CPU	PB112_I3CPU	PB9_I3CPU	PB8_I3CPU	PB6_I3CPU	PB2_I3CPU	Res.	PA7_I3CPU	PA6_I3CPU	PA1_I3CPU
	Reset value																					0	0	0	0	0	0	0	0		0	0	0
0xB4	PWR_I3CPUCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PH13_I3CPU	Res.	PG14_I3CPU	PG13_I3CPU	PG8_I3CPU	PG7_I3CPU	Res.	PD13_I3CPU	PD12_I3CPU	Res.	PC1_I3CPU	PC0_I3CPU
	Reset value																					0		0	0	0	0		0	0		0	0

Refer to *Section 2.3* for the register boundary addresses.

10 Reset and clock control (RCC)

10.1 RCC introduction

The reset and clock control (RCC) manages the different kinds of reset, and generates all clocks for the bus and peripherals.

10.2 RCC functional description

10.2.1 RCC pins and internal signals

The table below lists the RCC inputs and output signals connected to package pins or balls.

Table 93. RCC input/output signals connected to package pins or balls

Pin name	Pin type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO	O	Output clock for external devices
MCO2	O	Output clock 2 for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for SAI1 and ADF1

10.2.2 RCC reset functional description

There are three types of reset:

- a system reset
- a power reset
- a backup domain reset

Power reset

A power reset is generated when one of the following events occurs:

- a brownout reset (BOR)
- when exiting Standby mode
- when exiting Shutdown mode

A BOR sets all registers to their reset values except the ones in the backup domain.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, TAMP, WKUP, IWDG, and Standby/Shutdown mode control) are not impacted.

When exiting Shutdown mode, a brownout reset is generated, resetting all registers except those in the backup domain.

System reset

A system reset sets all registers to their reset values except the reset flags in RCC_CSR, and the registers in the backup domain.

A system reset is generated when one of the following events occurs:

- a low level on the NRST pin (external reset)
- a window watchdog event (WWDG reset)
- an independent watchdog event (IWDG reset)
- a software reset (SW reset)
- a low-power mode security reset
- an option-byte loader reset
- a brownout reset

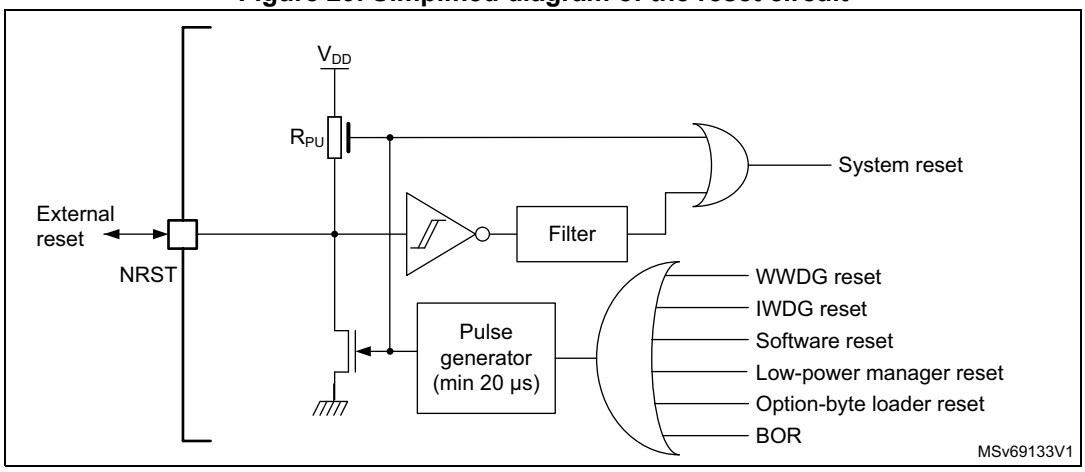
The reset source can be identified by checking the reset flags in RCC_CSR.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service routine vector is selected via the boot option bytes.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case of an internal reset, the internal pull-up R_{PU} is deactivated in order to save the power consumption through the pull-up resistor.

Figure 29. Simplified diagram of the reset circuit



- Software reset

  The SYSRESETREQ bit in Cortex-M33 application interrupt and reset control register must be set to force a software reset on the device.

- Low-power mode security reset

  To avoid that critical applications mistakenly enter a low-power mode, the following low-power mode security resets are available. If enabled in option bytes, the resets are generated in any of the following conditions:

  – Entering Standby mode: this type of reset is enabled by resetting NRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.

  – Entering Stop mode: this type of reset is enabled by resetting NRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

  – Entering Shutdown mode: this type of reset is enabled by resetting NRST_SHDW bit in user option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

  For further information on the user option bytes, refer to *Section 7.4.2: Option-byte programming*.

- Option byte loader reset

  The option byte loader reset is generated when the OBL_LAUNCH bit is set in FLASH_CR. This bit is used to launch the option byte loading by software.

Backup domain reset

The backup domain has two specific resets.

A backup domain reset is generated when one of the following events occurs:

- a software reset, triggered by setting the BDRST bit in RCC_BDCR
- a V_{DD} or V_{BAT} power on, if both supplies have previously been powered off

A backup domain reset affects the LSE oscillator, the RTC, the TAMP, the backup registers, and RCC_BDCR.

10.2.3 RCC clock functional description

Three different clock sources can be used to drive the system clock (SYSCLK):

- HSI16: high-speed internal 16 MHz RC oscillator clock
- MSIS: multi-speed internal RC oscillator clock, from 3 to 96 MHz
- HSE: high-speed external crystal or clock, from 4 to 50 MHz

The MSIS is used as system clock source after startup from reset, configured at 12 MHz.

The devices have the following additional clock sources:

- MSIK: multi-speed internal RC oscillator clock used for peripherals kernel clocks, from 3 to 96 MHz
- LSI: 32 kHz/250 Hz low-speed internal RC that drives the independent watchdog and optionally the RTC used for automatic wake-up from Stop and Standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)
- HSI48: internal 48 MHz RC that potentially drives the USB, the SDMMC and the RNG

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequency, the APB1, APB2, and APB3 domains. The maximum frequency of the AHB and APB domains is 96 MHz.

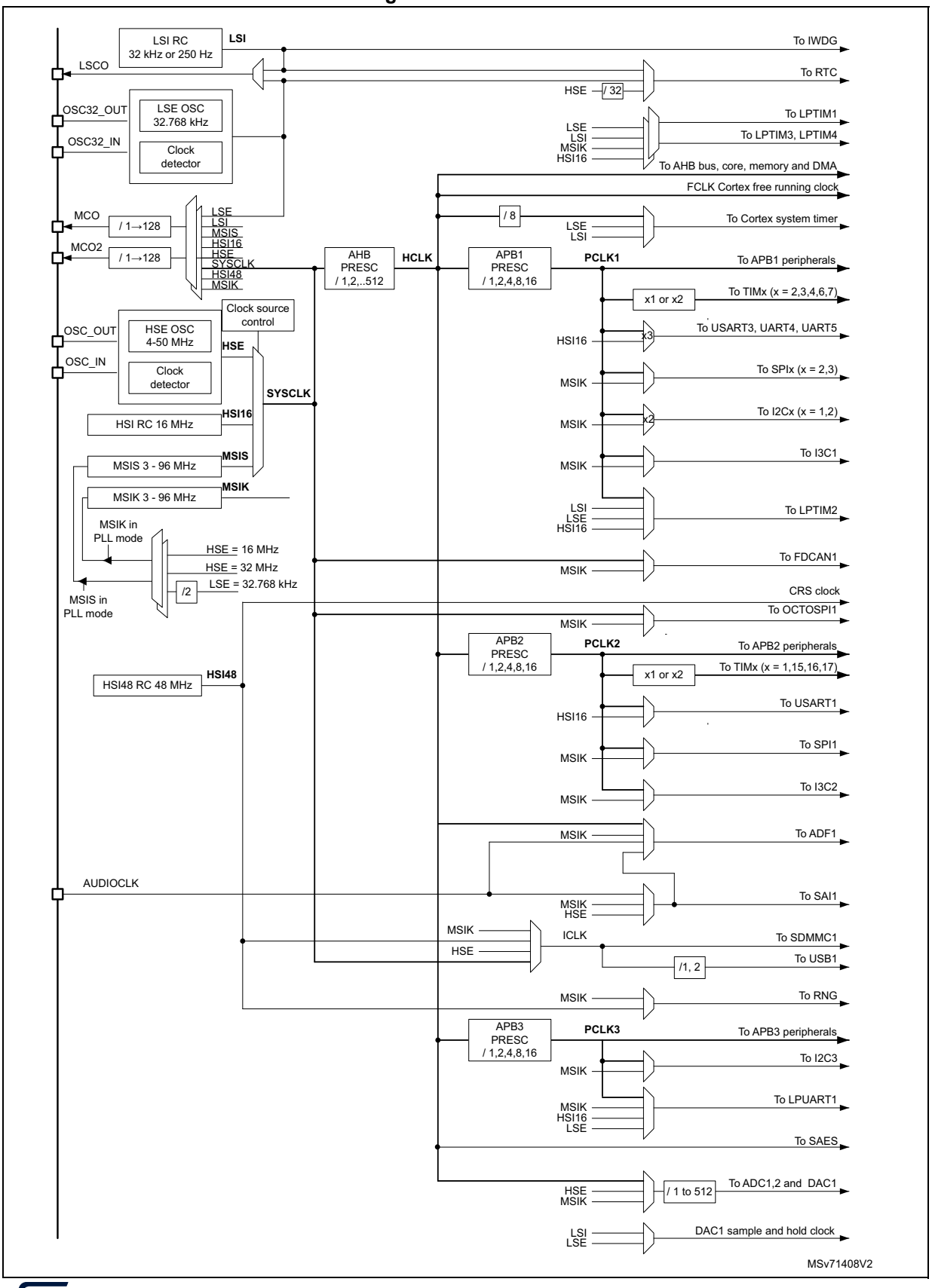
All peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2, or PCLK3), except the following ones that receive an independent kernel clock: USB, SDMMC1, RNG, ADCx (x = 1, 2), DAC1, U(S)ARTx (x = 1, 3 to 5), LPUART1, I2Cx (x = 1 to 3), I3Cx (x = 1, 2), SPIx (x = 1 to 3), OCTOSPI1, SAI1, ADF1, FDCAN1, LPTIMx (x = 1 to 4). Most of these kernel clocks can be selected by software from various sources, thanks to RCC_CCIPRx registers (x = 1 to 3).

In addition, the RTC kernel clock is selected by software in RCC_BDCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the core system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE, or LSI. The SysTick can work either with this clock or directly with the core clock (HCLK), configurable in the SysTick control and status register.

FCLK acts as Cortex-M33 free-running clock.

Figure 30. Clock tree



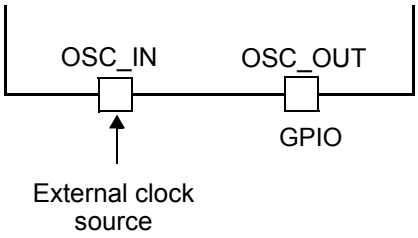
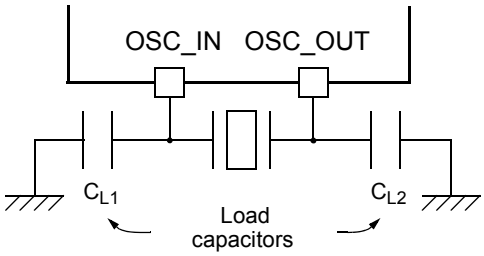
HSE clock

The high-speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Table 94. HSE/LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/ceramic resonators	

- External crystal/ceramic resonator (HSE crystal)

  The external oscillator has the advantage of producing a very accurate rate on the main clock.

  The associated hardware configuration is shown in *Table 94: HSE/LSE clock sources*. Refer to the electrical characteristics section of the datasheet for more details.

  The HSERDY flag in RCC_CR indicates if the HSE oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in RCC_CIER.

  The HSE crystal can be switched on and off using the HSEON bit in RCC_CR.

- External source (HSE bypass)

  In this mode, an external clock source must be provided. This mode is selected by setting the HSEBYP and HSEON bits in RCC_CR. The external clock signal with

~40 - 60 % duty cycle depending on the frequency (refer to the datasheet) must drive the OSC_IN pin while the OSC_OUT pin can be used a GPIO (see *Table 94: HSE/LSE clock sources*). The bypass mode is optimized for square input signals when HSEEXT = 1. If the input is a sine wave or triangle signal, HSEEXT must be kept at 0 (analog bypass mode).

HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator. However, even with calibration, the frequency is less accurate than an external crystal oscillator or ceramic resonator.

The HSI16 clock can be selected as system clock after wake-up from Stop modes (Stop 0, Stop 1, Stop 2, or Stop 3). Refer to RCC_CFGR1. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Clock security system (CSS)*.

- HSI16 calibration

  The RC oscillator frequencies may vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at T_A = 25°C.

  After reset, the factory calibration value is loaded in the HSICAL[11:0] bits in RCC_ICSCR3.

  If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The HSI16 frequency can be trimmed in the application using the HSITRIM[4:0] in RCC_ICSCR3.

  For more details on how to measure the HSI16 frequency variation, refer to *Internal/external clock measurement with TIM15/TIM16/TIM17*.

  The HSIRDY flag in RCC_CR indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

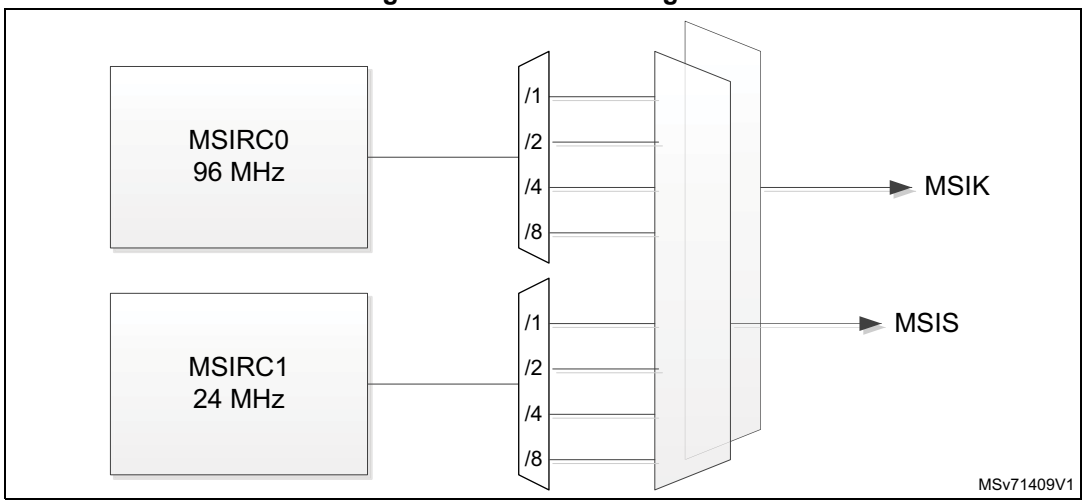
  The HSI16 RC can be switched on and off using the HSION bit in RCC_CR.

  The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Clock security system (CSS)*.

MSI (MSIS and MSIK) clocks

The MSI is made of two internal RC oscillators: MSIRC0 at 96 MHz and MSIRC1 at 24 MHz. Each oscillator feeds a prescaler providing a division by 1, 2, 4, or 8. Two output clocks are generated from these divided oscillators: MSIS, that can be selected as system clock, and MSIK, that can be selected by some peripherals as kernel clock.

Figure 31. MSI block diagram



For each output clock MSIS and MSIK, the oscillator source is selected using respectively MSISSEL and MSIKSEL, and division factor is selected using respectively MSISDIV[1:0] and MSIKDIV[1:0] in RCC_ICSCR1. A total of six different frequencies are available, generated from the two internal RCs, as shown in *Table 95*. 24 MHz and 12 MHz can be generated from both RCs.

In case no higher frequency is needed in the application, it is recommended to select MSIRC1 to get lower consumption.

In case 48 MHz or 96 MHz frequency is also needed in addition to 24 or 12 MHz, it is recommended to select MSIRC0, and generate both frequencies from the same oscillator rather than using both RCs, in order to save additional RC consumption.

Table 95. MSIS and MSIK ranges per internal MSIRCs (PLL mode disabled)

MSIRC0	MSIRC1
Range 0: 96 MHz	Range 4: 24 MHz
Range 1: 48 MHz	Range 5: 12 MHz
Range 2: 24 MHz	Range 6: 6 MHz
Range 3: 12 MHz	Range 7: 3 MHz

Note: Refer to datasheet for complete MSI frequency characteristics in MSI-mode and in PLL mode.

The MSIS clock is used as system clock after restart from reset, wake-up from Standby and Shutdown low-power modes. After restart from reset or when exiting Shutdown mode, the MSIS and MSIK frequencies are set to their default value 12 MHz, generated from MSIRC1. The selected oscillator is always MSIRC1 when exiting Standby mode, and the frequency range can be adjusted by software between 3, 6 or 12 MHz, using respectively the MSISDIVS[1:0] and MSIKDIVS[1:0] fields. Refer to RCC_CSR.

The MSIS clock can be selected as system clock after a wake-up from Stop mode (Stop 0, Stop 1, Stop 2, or Stop 3) depending on STOPWUCK in RCC_CFGR1. It can also be used

as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Clock security system (CSS)*.

The MSI oscillator has the advantage of providing a low-cost (no external components) low-power clock source. In addition, when used in PLL mode with the LSE or HSE, the MSI provides a very accurate clock source that can be used by the USB peripheral.

MSISRDY and MSIKRDY flags in RCC_CR indicate whether the MSIS and MSIK RC are stable or not. At startup, the MSIS and MSIK RC output clocks are not released until their respective bit is set by hardware. The MSIS and MSIK RC can be switched on and off by using the MSISON and MSIKON bits in RCC_CR.

- Hardware auto-calibration with LSE or HSE (PLL mode)

  When a 32.768 kHz, a 32 MHz or a 16 MHz external oscillator is present in the application, it is possible to configure MSIS, MSIK or both in a PLL mode. The long term accuracy of the MSI in PLL mode is the one of the external oscillator.

  Using the MSI in PLL mode with LSE provides a low-power solution to get an accurate high-speed clock.

  The MSIS is in PLL mode when the internal MSIRCx (x = 0,1) selected for MSIS is configured in PLL mode. The MSIK is in PLL mode when the internal MSIRCx (x = 0,1) selected for MSIK is configured in PLL mode.

  Either MSIS or MSIK, with MSIRC1 selected, must be enabled and ready before enabling MSIRC1 PLL mode, and in the same way either MSIS or MSIK, with MSIRC0 selected, must be enabled and ready before enabling MSIRC0 PLL mode.

  When LSE or HSE is used for PLL mode, this oscillator must be enabled and ready before selecting it as input for MSIRCx PLL mode.

  MSIRCx PLL mode is configured to operate with LSE when the corresponding MSIPLLxSEL = 0, or with HSE at 32 MHz or 16 MHz when the corresponding MSIPLLxSEL = 1 in RCC_ICSCR1. In case HSE is at 32 MHz, it must be divided by 2 by setting MSIHSINDIV bit.

  Once input clock is selected, the MSIRCx PLL mode is enabled by setting the corresponding MSIPLLxEN bit in RCC_CR. Refer to *Table 96* for an overview of available frequencies in this mode.

  If the PLL input clock (LSE or HSE) pulses are stopped or temporarily disturbed, the MSI PLL mode is automatically unlocked, and the MSI accuracy is consequently degraded. If enabled in RCC_CIER, an interrupt is generated. The MSI PLL mode with LSE unlock interrupt can wake up from Stop 0, Stop 1 and Stop 2 modes. To reconfigure the MSI in PLL mode, the MSIPLLxEN (x = 0,1) bit in RCC_CR must be cleared and set again.

  When MSIRC1 is used in PLL mode, the reached frequency can be selected thanks to MSIPLL1N[1:0] bitfield in the RCC_ICSR1. Those additional frequencies are useful for audio applications.

Table 96. MSIS and MSIK frequencies of MSIRC1 in PLL mode

MSI range	MSISDIV[1:0] or MSIKDIV[1:0]	PLL range MSIPLL1N[1:0]	MSIPLL1SEL = 0: LSE	MSIPLL1SEL = 1: HSE or HSE/2
Range 0: MSISSEL = 0 or MSIKSEL = 0	00: MSIRC0/1	XX	96.010 MHz	96.0 MHz
	01: MSIRC0/2	XX	48.005 MHz	48.0 MHz
	10: MSIRC0/4	XX	24.003 MHz	24.0 MHz
	11: MSIRC0/8	XX	12.001 MHz	12.0 MHz
Range 1: MSISSEL = 1 or MSIKSEL = 1	00: MSIRC1/1	0X	23.986 MHz	24.016 MHz
		10	22.577 MHz	22.581 MHz
		11	24.576 MHz	24.577 MHz
	01: MSIRC1/2	0X	11.993 MHz	12.008 MHz
		10	11.289 MHz	11.290 MHz
		11	12.288 MHz	12.289 MHz
	10: MSIRC1/4	0X	5.997 MHz	6.004 MHz
		10	5.644 MHz	5.645 MHz
		11	6.144 MHz	6.144 MHz
	11: MSIRC1/8	0X	2.998 MHz	3.002 MHz
		10	2.822 MHz	2.823 MHz
		11	3.072 MHz	3.072 MHz

- MSI PLL mode stabilization time

  When MSIPLLxEN (x = 0,1) = 1, the final accuracy after enabling the MSI (by writing MSISON = 1 or MSIKON = 1 or following a peripheral clock request in Stop mode) is reached after a stabilization time t_{STAB}(MSI) when the corresponding MSIPLLxFAST = 0. This stabilization time is needed even if the input clock (LSE or HSE) is kept enable. Refer to datasheet for t_{STAB}(MSI) value.

  If MSIPLLxEN = 1 with MSIPLLxFAST = 1 , the MSI oscillator is kept powered on when a request to switch it off is received (either by writing MSISON = 0 and MSIKON = 0, or because no peripheral requests this clock in Stop mode). In this case the MSI PLL mode accuracy is kept when the MSI is switched on again, providing that the t_{STAB}(MSI) stabilization time is reached before switching off the MSI. This mode can be used for autonomous peripherals requiring accuracy in Stop mode, with an extra consumption as the oscillator remains powered on, but gated off when disabled.

- Software calibration

  The MSIRCx (x = 0,1) oscillators frequency may vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at an ambient temperature, T_A = 25 °C. After reset, the factory calibration value is loaded in the MSICALx[5:0] bits (x = 0,1) in RCC_ICSCR1. If the application is subject to voltage or temperature variations, this may affect the RC

oscillator speed. The MSIRCx frequency can be trimmed in the application by using the MSITRIMx[5:0] bits (x = 0,1) in RCC_ICSCR2.

Note: The final accuracy after applying the calibration value is reached after a stabilization time. This stabilization time is needed after reset of exiting Standby or Shutdown mode. It is also needed when switching from PLL mode to normal mode.

The hardware auto-calibration with LSE or HSE must not be used in conjunction with software calibration.

For more details on how to measure the MSI frequency variation, refer to *Internal/external clock measurement with TIM15/TIM16/TIM17*.

HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used directly for USB and for random number generator (RNG) as well as SDMMC.

The internal 48 MHz RC oscillator is mainly dedicated to provide a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE or an external signal to automatically and quickly adjust the oscillator frequency on-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency that is subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral, refer to *Section 11: Clock recovery system (CRS)*.

The HSI48RDY flag in RCC_CR indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in RCC_CR.

LSE clock

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in RCC_BDCR. If the LSE is used by other peripherals of functions than RTC, TAMP, and LSECSS, the LSESYSEN bit must be also be set in RCC_BDCR (refer to *LSE when used by peripherals other than RTC/TAMP, and RCC functions*).

The crystal oscillator driving strength is configured using the LSEDRV[1:0] bits, according to crystal specification, to obtain the best compromise between robustness and short startup time on one side and low-power-consumption on the other side. The LSE drive must be programmed before enabling the LSE.

The LSERDY flag in RCC_BDCR indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in RCC_CIER.

- External source (LSE bypass)

  In this mode, an external clock source must be provided. This mode is selected by setting the LSEBYP and LSEON bits in RCC_BDCR. The external clock signal (square, sinus or triangle) with ~50 % duty cycle, must drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO (see *Table 94: HSE/LSE clock sources*).

- LSE when used by peripherals other than RTC/TAMP, and RCC functions

  By default, when enabled, the LSE is sent only to RTC and TAMP (assuming that RTCSEL = 01).

  If the LSE is needed for other peripherals (such as peripheral clock or trigger source), or if the LSE is used by a RCC function (such as LSCO, MCO, MSI PLL mode), the sequence below must be done:

  a) Set the LSEON in RCC_BDCR, and wait the LSE clock ready bit (LSERDY = 1 in RCC_BDCR).

  b) Set the LSESYSEN bit in RCC_BDCR.

  c) Wait the LSESYS clock is ready (LSESYSRDY = 1 in RCC_BDCR).

  The LSE consumption is increased when LSESYSEN = 1.

LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby modes for the IWDG and RTC/TAMP. The clock frequency is either 32 kHz or 250 Hz, depending on the LSIPREDIV bit in RCC_CSR. Setting LSIPREDIV allows a lower consumption (refer to the electrical characteristics section of the datasheet for more details).

When the IWDG is enabled or when the RTC or TAMP is clocked by the LSI, the LSIPREDIV cannot be changed anymore.

The LSI RC can be switched on and off using the LSION bit in RCC_CSR.

The LSIRDY flag in RCC_CSR indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in RCC_CIER.

System clock (SYSCLK) selection

Three different clock sources can be used to drive the system clock (SYSCLK):

- MSIS oscillator
- HSI16 oscillator
- HSE oscillator

The system clock maximum frequency is 96 MHz. After a system reset, the MSIS oscillator, at 12 MHz, is selected as system clock. When a clock source is used as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clocks are ready and which clock is currently used as a system clock.

Table 97 gives the different bus frequencies depending on the product voltage range.

The EPOD (embedded power distribution) booster must be enabled when the system clock frequency is above 24 MHz in range 1 and range 2 (see *Dynamic voltage scaling management*). Before enabling this booster, its input clock must be selected and divided so that its frequency is between 3 and 16 MHz. The division factor is forced by hardware when the selected booster clock is MSIS. The booster must be enabled before entering Stop mode, with MSIS as the selected clock, if the wake-up clock is MSIS at 48 MHz.

Caution: The booster clock source must be kept enabled as long as the booster is enabled in Run and Sleep modes.

Table 97. Bus maximum frequency

Product voltage range		AHB1/AHB2/APB1/APB2/APB3
Range 1	EPOD booster enabled	96 MHz
	EPOD booster disabled	24 MHz
Range 2	EPOD booster enabled	48 MHz
	EPOD booster disabled	24 MHz

Clock source frequency versus voltage scaling

The table below gives the different clock source frequencies depending on the product voltage range.

Table 98. Clock source maximum frequency

Product voltage range	Clock frequency			
	MSIS MSIK	HSI16	HSI48	HSE
Range 1	All ranges	Allowed	Allowed	50 MHz
Range 2	Range 1 to 7	Allowed	Allowed	48 MHz

Clock security system (CSS)

The CSS can be activated by software. In this case, the clock detector is enabled after the HSE oscillator wake-up time, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled. A clock failure event is sent to some timers break input and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI). This allows the MCU to perform rescue operations. The CSSI is linked to the Cortex-M33 NMI (non-maskable interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSSF occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSSF pending bit is cleared. As a consequence, in the NMI ISR, the user must clear the CSSF by setting the CSSC bit in RCC_CICR.

If the HSE oscillator is used as the system clock, a detected failure causes a switch of the system clock to the MSIS or the HSI16 oscillator depending on the STOPWUCK configuration in RCC_CFGR1, and the disabling of the HSE oscillator.

Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in RCC_BDCR. LSECSSON must be written after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after the RTC/TAMP clock has been selected by RTCSEL.

The CSS on LSE is working in all modes including VBAT. It works also under system reset (excluding power-on reset). If enabled in RCC_CIER, an interrupt is generated. The CSS on

LSE interrupt can wake up from all low-power modes. Refer to *Section 9: Power control (PWR)* for initialization sequence to wake-up from Stop 3 and Standby with CSS on LSE detection.

The clock security system on LSE detects when the LSE disappears or in case of over frequency. In addition, the glitches on LSE can be filtered by setting LSEGFON. LSEGFON must be written when the LSE is disabled (LSEON = 0 and LSERDY = 0).

If a failure is detected on the external 32 kHz oscillator (LSECSSD = 1), the LSE clock is no longer supplied to the RTC/TAMP but no hardware action is made to the registers. If the MSI was in PLL mode, this mode is disabled.

The CSS on LSE detection event is connected to the internal tamper 3 of the TAMP peripheral.

In case of CSS on LSE detection event (LSECSSD = 1 in RCC_BDCR), the software can decide either to keep on working with LSE, or to change RTC/TAMP clock source.

In order to enable again RTC and TAMP clocking with LSE clock, the software must clear the LSECSSON bit in RCC_BDCR. Disabling the CSS on LSE (LSECSSON = 0) also clears the LSE failure detection flag (LSECSSD), and allows LSE clock to be propagated to the RTC and TAMP.

In order to change the RTC/TAMP clock source, the software must first enable the target oscillator and wait for it to be ready, then change the RTCSEL[1:0]. Then the CSS on LSE should be disabled (LSECSSON = 0) as well as LSE oscillator (LSEON = 0).

ADC and DAC clocks

The ADC and DAC kernel clock source is selected thanks to ADCDACSEL[1:0] in RCC_CCIPR2. The selected clock source is divided thanks to ADCDACPRE[3:0] bitfield, from division by 1 to division by 512. The ADC clock ratio must be around 50 %. For this reason, the AHB clock, when selected as ADC clock, must not be divided with HPRE prescaler.

If the application requires that the ADC or DAC is precisely triggered by a TIMx timer without any uncertainty, the HCLK must be selected as ADC and DAC kernel clock source. The other clock sources are asynchronous to TIMx timers therefore an uncertainty of the trigger instant is added by the resynchronization between the two clock domains. The LPTIMx timers are also asynchronous.

The DAC requires an additional low-power clock (LSI or LSE) to operate in sample and hold mode, available in Stop mode. This clock is selected with DAC1SHSEL in RCC_CCIPR2.

RTC and TAMP clock

The RTCCLK clock source is used by RTC and TAMP, and can be either the HSE / 32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in RCC_BDCR. This selection cannot be modified without resetting the backup domain, except in case of CSS on LSE failure detection. The system must always be configured so as to get a PCLK frequency greater then or equal to the RTCCLK frequency for a proper operation of the RTC. The TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 44: Tamper and backup registers (TAMP)*).

The LSE is in the backup domain, whereas the LSI and HSE clock are not. Consequently:

- If LSE is selected as RTC and TAMP clock, these peripherals continue to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If the LSI or HSE clock divided by a prescaler is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the core domain). Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

If the LSE is needed only for the RTC or TAMP, LSESYSEN must be kept at reset value to get the lowest consumption.

Timer clock

The timer clock frequencies are automatically defined by hardware.

There are two cases:

- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency.
- Otherwise, they are set to twice (×2) the APB domain frequency.

Watchdog clock

If the independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced on and cannot be disabled. After the LSI oscillator temporization, the LSI 32 kHz clock is provided to the IWDG.

Clock-out capability

- **MCO**

  The microcontroller clock output (MCO) capability allows a clock to be output onto two external MCO and MCO2 pins. One of the following clock signals can be selected as MCO/MCO2 clock.

  – LSI
  – LSE
  – SYSCLK
  – HSI16
  – HSI48
  – HSE
  – MSIS
  – MSIK

  The selection is controlled by the MCOSEL[3:0] and MCO2SEL[3:0] bits in RCC_CFGR1. The selected clock can be divided with the MCOPRE[2:0] and MCO2PRE[2:0] field in RCC_CFGR1.

  The MCO clock output requires the corresponding alternate function selected on the MCO pin.

- **LSCO**

  Another output (LSCO) allows one of the low-speed clocks below to be output onto the external LSCO pin:

  – LSI

  – LSE

  This output remains available in all Stop, Standby, and Shutdown modes. This output is not available in V_{BAT} mode. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN in RCC_BDCR.

Internal/external clock measurement with TIM15/TIM16/TIM17

The frequency of all on-board clock sources can be indirectly measured by mean of the TIM15, TIM16, or TIM17 channel 1 input capture and LPTIM1 or LPTIM2 channel 2 input capture.

- HSI16 and MSI calibration using LSE

  The primary purpose of connecting the LSE to the channel 1 input capture of TIM15, TIM16, and TIM17, and to the channel 2 input capture of LPTIM1, is to be able to precisely measure the HSI16 and MSI system clocks (for this, either HSI16 or MSIS must be used as system clock source). The number of HSI16 (MSIS respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm), the internal clock frequency can be determined with the same resolution, and the source can be trimmed to compensate the manufacturing, process, temperature and/or voltage related frequency deviations.

  The four oscillators of MSI and HSI16 oscillator have dedicated user-accessible calibration bits for this purpose.

  The basic concept consists in providing a relative measurement (such as HSI16/LSE ratio). The precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

  Note: When the LSE or HSE is available, the MSI can be automatically trimmed by LSE or HSE using the PLL mode.

- HSI16 and MSI calibration using HSE

  If the HSE is available, it can be used as system clock and the timer input capture must be connected either to MSI (divided by 1024 or by 4) or to HSI/256. The TIM16 and TIM17 channel 1 input capture, as well and the LPTIM2 input capture 2, are connected to the divided oscillator only when TIMICSEL[2:0] is different from 0xx in RCC_CCIPR1.

  Considering that the timers counter is 16-bit, and that the ratio between HSE and the input capture signal must be the highest possible, a division by 1024 must be selected when the MSIRC0, MSIRC1 or MSIRC2 is measured, and a division by 4 when the MSIRC4 is measured.

- LSI calibration

  The calibration of the LSI follows the same principle, but changing the reference clock. The LSI clock must be connected to the channel 1 input capture of the TIM16 or TIM17, or to the channel 2 input capture of the LPTIM1. Then defining the HSE as system

clock source, the number of its clock counts between consecutive edges of the LSI signal, provides a measure of the internal low-speed clock period.

The basic concept consists in providing a relative measurement (such as the HSE/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement is.

Peripherals clock gating and autonomous mode

- Peripherals clock gating in Run mode

  Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

  When the peripheral clock is not active, read or write accesses to the peripheral registers are not supported.

  The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, there the clock is active after 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.

- Peripherals clock gating in Sleep mode

  When a peripheral is enabled, its clock can be automatically gated off when the device is in Sleep mode, by clearing the peripheral SLPEN bit in RCC_AHBxSLPENR and RCC_APBxSLPENR. Both EN and SLPEN bit of the peripheral must be set to keep the clock on in Sleep mode.

- Peripherals clock gating and autonomous mode in Stop 0/1/2 modes

  When a peripheral is enabled, its clock can be automatically gated off when the device is in Stop mode, by clearing the peripheral STPEN bit in RCC_AHBxSTPENR and RCC_APBxSTPENR. All EN, SLPEN and STPEN bits of the peripheral must be set to keep the clock on in Stop mode.

  The peripherals listed hereafter support autonomous mode in Stop 0 and Stop 1 modes: U(S)ARTx (x = 1, 3 to 5), LPUART1, SPIx (x = 1 to 3), I2Cx (x = 1 to 3), I3Cx (x = 1,2), LPTIMx (x = 1 to 4), ADF1, DAC1. Only LPUART1, I2C3, LPTIM1, LPTIM3, and LPTIM4 support autonomous mode in Stop 2 mode. These peripherals are able to generate a kernel clock request and a AHB/APB bus clock request when they need, in order to operate and update their status register even in Stop mode. Depending on the peripheral configuration, either a DMA request or an interrupt can be associated to the peripheral event.

  Upon an AHB or APB bus clock request from an autonomous peripheral, either MSIS or HSI16 oscillator is woken up, depending on the oscillator selected by STOPWUCK in RCC_CFGR1.

  If the autonomous peripheral is configured with DMA requests enabled, a data transfer is performed thanks to the AHB/APB clock. The bus clocks as well as the oscillator (HSI16 or MSIS) are automatically switched off as soon as the transfer is finished, if no other peripheral requests it. The device automatically goes back in Stop mode.

  If the autonomous peripheral is configured with interrupt enabled, the interrupt wakes up the device into Run mode.

  A peripheral autonomous mode is enabled in Stop 0, Stop 1, or Stop 2 modes if all peripheral EN, SLPEN and STPEN bits are set.

  If an autonomous peripheral requests its kernel clock in Stop 0, Stop 1, or Stop 2 mode, the internal oscillator (HSI16 or MSIK) is woken up if it was off, and the kernel clock is

propagated only to the peripheral requesting it. When the peripheral releases its kernel clock request, the HSI16 or MSI is switched off if no other peripheral requests it.

If an autonomous peripheral requests its AHB or APB bus clock in Stop 0, Stop 1 or Stop 2 mode, the internal oscillator (HSI16 or MSIS depending on STOPWUCK value in RCC_CFGR1) is woken up if it was off, and the system clock is propagated to all peripherals configured with EN, SLPEN and STPEN bits. Therefore, the peripheral STPEN bit should be cleared to reduce consumption if the peripheral is not used in Stop mode.

Caution: The EN, SLPEN and STPEN bits of the peripheral must be set to allow the generation of an interrupt capable to wake up the device from Stop 0, Stop 1, and Stop 2 modes. This is not necessary when the peripheral wake-up interrupt is generated though the EXTI.

Note: MSIK or HSI16 can be forced to remain ON in Stop 0, Stop 1 or Stop 2 mode, by configuring MSIKERON or HSIKERON is the RCC_CR. In this case, the oscillator is propagated only to the kernel clock of the enabled autonomous peripherals with this oscillator selected as kernel clock. This allows the peripheral baudrates or conversion rates increase, as there is no need to wait for the oscillator wake-up time when the peripheral requests its kernel clock.

The LSE or LSI selected as peripheral kernel clock remains always ON in Stop modes.

10.2.4 RCC security and privilege functional description

RCC TrustZone security protection modes

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register, the RCC is able to secure RCC configuration and status bits from being modified by nonsecure accesses.

This is configured through RCC_SECCFGR to prevent nonsecure access to read or modify the following features:

- HSE, HSE-CSS, HSI, MSI, LSI, LSE, LSE-CSS, LSCO, HSI48 configuration and status bits
- AHB and APB prescalers configuration and status bits
- System clock (SYSCLK) and ICLK source clock selection and status bits
- MCO/MCO2 clock output configuration and STOPWUCK and STOPKERWUCK bit
- Remove reset flag RMVF configuration

If SPRIV is set in RCC_PRIVCFGR, RCC_SECCFGR can be written only by secure and privileged access. If SPRIV is cleared in RCC_PRIVCFGR, RCC_SECCFGR can be written only by secure access, privileged or unprivileged.

RCC_SECCFGR can be read by secure, nonsecure, privileged and unprivileged access.

When a peripheral is configured as secure, its related clock, reset, clock source selection and clock enable during low-power modes control bits, are also secure in RCC_AHBxENR, RCC_APBxENR, RCC_AHBxSLPENR, RCC_APBxSLPENR, RCC_AHBxSTPENR, RCC_APBxSTPENR, RCC_CCIPR1, RCC_CCIPR2, RCC_CCIPR3, and RCC_BDCR.

BDRST in RCC_BDCR is secure when at least one function is secure in RTC or TAMP.

If several peripherals share same RCC resources, these resources are secure when at least one peripheral is configured as secured.

A peripheral is secure when:

- For securable peripherals by TZSC (TrustZone security controller), the SEC security bit corresponding to this peripheral is set in the GTZC TZSC secure configuration registers.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 99 summarizes the RCC secured bits following the security configuration bit in RCC_SECCFGR.

When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and nonsecure bits:

- Secured bits: read and write operations are only allowed by a secure access. Nonsecure read returns 0 and write accesses are ignored. No illegal access event is generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure accesses.
- A nonsecure write access to RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in GTZC TZIC registers. RCC_SECCFGR can be read by secure or nonsecure access.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR register), all registers are nonsecure. RCC_SECCFGR write accesses are ignored.

Table 99. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL[11:0], HSITRIM[4:0]	RCC_ICSCR3
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT	RCC_CR
	HSERDYIE, HSECSSF	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR

Table 99. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
MSISEC	MSISON, MSIKERON, MSISRDY, MSPLL0EN, MSIPLL1EN, MSIPLL0RDY, MSIPLL1RDY, MSIKON, MSIKRDY, MSIPLL0FAST, MSIPLL1FAST	RCC_CR
	MSISSEL, MSISDIV[1:0], MSIKSEL, MSIKDIV[1:0], MSIPLL1N[1:0], MSIPLL0SEL, MSIPLL1SEL, MSIHSINDIV, MSIRGSEL, MSIBIAS, MSICAL0[5:0], MSICAL1[5:0]	RCC_ICSCR1
	MSITRIM0[5:0], MSITRIM1[5:0]	RCC_ICSCR2
	MSIPLL0RDYIE, MSIPLL1RDYIE, MSIPLLUIE, MSIPLLHSUIE	RCC_CIER
	MSIPLL0RDYIF, MSIPLL1RDYIF, MSIPLLUIF, MSIPLLHSUIF	RCC_CIFR
	MSISRDYC, MSIKRDYC, MSIPLL0RDYC, MSIPLL1RDYC, MSIPLLUC, MSIPLLHSUC	RCC_CICR
	MSISDIVS[1:0], MSIKDIVS[1:0]	RCC_CSR
LSISEC	LSCOSEL, LSCOEN	RCC_BDCR
	LSION, LSIRDY, LSIPREDI	RCC_CSR
	LSIRDYIE	RCC_CIER
	LSIRDYIF	RCC_CIFR
	LSIRDYC	RCC_CICR
LSESEC	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSEGFON, LSESYSRDY, LSESYSEN, LSCOSEL, LSCOEN	RCC_BDCR
	LSERDYIE	RCC_CIER
	LSERDYF	RCC_CIFR
	LSERDYC	RCC_CICR
SYSCLKSEC	SW[1:0], SWS[1:0], STOPWUCK, STOPKERWUCK, MCOSEL[3:0], MCOPRE[2:0], MCO2SEL[3:0],MCO2PRE[2:0]	RCC_CFGR1
	SYSTICKSEL[1:0]	RCC_CCIPR1
	R1EN, R2EN, BOOSTEN	PWR_VOSR
PRESCSEC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0]	RCC_CFGR2
	PPRE3[2:0]	RCC_CFGR3
BOOSTSEC	BOOSTSEL[1:0], BOOSTDIV[3:0]	RCC_CFGR4
HSI48SEC(1)	HSI48ON, HSI48RDY	RCC_CR
	HSI48CAL[8:0]	RCC_CRRCR
	HSI48RDYE	RCC_CIER
	HSI48RDYF	RCC_CIFR
	HSI48RDYC	RCC_CICR
ICLKSEC	ICLKSEL[1:0]	RCC_CCIPR1
RMVFSEC	RMVF	RCC_CSR

1. TRIM field of the HSI48 is located in CRS peripheral. Be sure to secure it using CRSSEC bit in GTZC TZSC secure configuration register 1.

RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access except RCC_PRIVCFGR that can be written with privileged access only. RCC_PRIVCFGR can be read by secure and nonsecure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all RCC secure functions (as defined by RCC_SECCFGR or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC secure bits is possible only with privileged access, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged access except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged access only, secure or nonsecure. This bit configures the privileged access of all RCC nonsecure functions (as defined by RCC_SECCFGR, or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC nonsecure bits is possible only with privileged access.
- The RCC nonsecure bits can be read only with privileged access except RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

10.3 RCC in low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (flash memory, caches and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

  Stop modes (Stop 0, Stop 1, Stop 2, Stop 3) stop all the clocks in the core domain and disable the HSI16, HSI48, MSI and HSE oscillators. However, HSI16 or MSI can be switched ON if the peripheral requests it for autonomous mode purpose (in Stop 0, Stop 1 and Stop 2 only), or to generate a wake-up interrupt (Stop 0, Stop 1 or Stop 2 only) (see *Peripherals clock gating and autonomous mode* for more details). LSI and LSE remain active in Stop modes.
- Standby and Shutdown modes stop all the clocks in the core domain and disable the HSI16, HSI48, MSI and HSE oscillators.

The CPU DeepSleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in DBGMCU_CR.

When exiting Stop modes (Stop 0, Stop 1, Stop 2, or Stop 3), the system clock is either MSIS or HSI16, depending on the software configuration of STOPWUCK in RCC_CFGR1. The frequency (range and user trim) of the MSIS and MSIK oscillators is the one configured before entering Stop mode, except if above 48 MHz. In this case, the MSIS or MSIK range is the 48 MHz range. The user trim of HSI16 is kept. If MSIRCx (x = 0,1) is in PLL mode before entering Stop mode with MSIPLLxFAST = 0, the PLL mode stabilization time must be waited for after wake-up even if the LSE was kept ON during the Stop mode. The PLL mode accuracy is kept after wake-up from Stop 0, Stop 1 or Stop 2 mode without stabilization time if MSIPLLxFAST = 1. The MSIPLLxFAST bit has no effect when exiting Stop 3 mode.

The other internal oscillator can be automatically woken up in addition to the one used by the system clock, in order to avoid waiting for the other oscillator wake-up time when the device is back in Run mode. This is done thanks to STOPKERWUCK in RCC_CFGR1.

When leaving Standby and Shutdown modes, the system clock is MSIS. The MSIS and MSIK frequency at wake-up from Standby mode is configured with MSISDIVS[1:0] and MSIKDIVS[1:0] in RCC_CSR, from 3 to 12 MHz. The MSI frequency at wake-up from Shutdown mode is 12 MHz. The user trim is lost.

If a flash memory programming operation is ongoing, Stop, Standby, or Shutdown mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, Stop, Standby, or Shutdown mode entry is delayed until the APB access is finished. If an autonomous peripheral generates a system clock request, Stop, Standby or Shutdown mode entry is delayed until the system clock request is released.

10.4 RCC interrupts

The following table summarizes the interrupt sources and the way to control them.

Table 100. Interrupt sources and control

Interrupt event flag	Description	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby, Shutdown modes
Interrupt vector: RCC						
LSIRDYF	LSI ready	LSIRDYIE = 0 LSISEC = 0	LSIRDYC = 1	Yes	No	No
LSERDYF	LSE ready	LSERDYIE = 0 LSESEC = 0	LSERDYC = 1	Yes	No	No
HSIDRYF	HSI ready	HSIDRYIE = 0 HSISEC = 0	HSIRDYC = 1	Yes	No	No
HSERDYF	HSE ready	HSERDYIE = 0 HSESEC = 0	HSERDYC = 1	Yes	No	No
MSISRDYF	MSIS ready	MSISRDYIE = 0 MSISEC = 0	MSISRDYC = 1	Yes	No	No
MSIKRDYF	MSIK ready	MSIKRDYIE = 0 MSISEC = 0	MSIKRDYC = 1	Yes	No	No

Table 100. Interrupt sources and control (continued)

Interrupt event flag	Description	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby, Shutdown modes
MSIPLL0RDYF	MSIRC0 PLL mode ready	MSIPLL0RDYIE = 0 MSISEC = 0	MSIPLL0RDYC = 1	Yes	No	No
MSIPLL1RDYF	MSIRC1 PLL mode ready	MSIPLL1RDYIE = 0 MSISEC = 0	MSIPLL1RDYC = 1	Yes	No	No
MSIPLLUF	MSI PLL mode with LSE unlock	MSIPLLUIE = 0 MSISEC = 0	MSIPLLUC = 1	Yes	Yes(1)	No
MSIPLLHSUF	MSI PLL mode with HSE unlock	MSIPLLHSUIE = 0 MSISEC = 0	MSIPLLHSUC = 1	Yes	No	No
LSECSSF	LSE clock security system detection	LSECSSIE = 0 LSESEC = 0	LSECSSC = 1	Yes	Yes	Yes
HSI48RDYF	HSI48 ready	HSI48RDYIE = 0 HSI48SEC = 0	HSI48RDYC = 1	Yes	No	No
Interrupt vector: RCC_S(2)						
LSIRDYF	LSI ready	LSIRDYIE = 1 LSISEC = 1	LSIRDYC = 1	Yes	No	No
LSERDYF	LSE ready	LSERDYIE = 1 LSESEC = 1	LSERDYC = 1	Yes	No	No
HSIDRYF	HSI ready	HSIDRYIE = 1 HSISEC = 1	HSIRDYC = 1	Yes	No	No
HSERDYF	HSE ready	HSERDYIE = 1 HSESEC = 1	HSERDYC = 1	Yes	No	No
MSISRDYF	MSIS ready	MSISRDYIE = 1 MSISEC = 1	MSISRDYC = 1	Yes	No	No
MSIKRDYF	MSIK ready	MSIKRDYIE = 1 MSISEC = 1	MSIKRDYC = 1	Yes	No	No
MSIPLL0RDYF	MSIRC0 PLL mode ready	MSIPLL0RDYIE = 1 MSISEC = 1	MSIPLL0RDYC = 1	Yes	No	No
MSIPLL1RDYF	MSIRC1 PLL mode ready	MSIPLL1RDYIE = 1 MSISEC = 1	MSIPLL1RDYC = 1	Yes	No	No
MSIPLLUF	MSI PLL mode with LSE unlock	MSIPLLUIE = 1 MSISEC = 1	MSIPLLUC = 1	Yes	Yes	No
MSIPLLHSUF	MSI PLL mode with HSE unlock	MSIPLLHSUIE = 1 MSISEC = 1	MSIPLLHSUC = 1	Yes	No	No
LSECSSF	LSE clock security system detection	LSECSSIE = 1 LSESEC = 1	LSECSSC = 1	Yes	Yes	Yes
HSI48RDYF	HSI48 ready	HSI48RDYIE = 1 HSI48SEC = 1	HSI48RDYC = 1	Yes	No	No
Interrupt vector: NMI						
HSECSSF	HSE CSS failure	- (3)	HSECSSC = 1	Yes	No	No

1. Except Stop 3 mode
2. The RCC secure interrupt vector is used only when TrustZone is enabled.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

10.5 RCC registers

10.5.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 001D

HSEBYP and HSEEXT are cleared upon power-on reset. They are not affected upon other types of reset.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON
											rw	rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSI48RDY	HSI48ON	HSIRDY	HSIKERON	HSION	MSIPLL0RDY	MSIPLL1RDY	MSIPLL0FAST	MSIPLL1FAST	MSIPLL0EN	MSIPLL1EN	MSIKRDY	MSIKON	MSISRDY	MSIKERON	MSISON
r	rw	r	rw	rw	r	r	rw	rw	rw	rw	r	rw	r	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HSEEXT**: HSE external clock bypass mode

Used to select the external clock mode in bypass mode. External clock mode must be configured with HSEON bit to be used by the device. This bit can be written only if the HSE oscillator is disabled. This bit is active only if the HSE bypass mode is enabled.

0: External HSE clock analog mode

1: External HSE clock digital mode (through I/O Schmitt trigger)

Bit 19 **HSECSSON**: Clock security system enable

Used to enable the clock security system. When HSECSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if a HSE clock failure is detected. This bit is set only and is cleared by reset.

0: Clock security system OFF (clock detector OFF)

1: Clock security system ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE crystal oscillator bypass

Used to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled (HSEON = 0 and HSERDY = 0).

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

Note: Once the HSEON bit is cleared, HSERDY goes low after six HSE clock cycles.

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software. Cleared by hardware to stop the HSE oscillator when entering Stop, Standby, or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

Bit 15 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that the HSI48 oscillator is stable. This bit is set only when HSI48 is enabled by software by setting HSI48ON.

0: HSI48 oscillator not ready

1: HSI48 oscillator ready

Bit 14 **HSI48ON**: HSI48 clock enable

Set and cleared by software. Cleared by hardware to stop the HSI48 when entering in Stop, Standby, or Shutdown mode.

0: HSI48 oscillator OFF

1: HSI48 oscillator ON

Bit 13 **HSIRDY**: HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable after HSION is set.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

Note: When the HSI is enabled due to HSIKERON or to peripheral clock request, the HSIRDY remains cleared. Once the HSION bit is cleared, HSIRDY goes low after six HSI16 clock cycles.

Bit 12 **HSIKERON**: HSI16 enable for some peripheral kernels

Used to force HSI16 ON even in Stop modes. Keeping the HSI16 ON in Stop mode allows the communication speed not to be reduced by the HSI16 startup time. This bit has no effect on HSION value (refer to *Peripherals clock gating and autonomous mode* for more details). The HSIKERON must be configured at 0 before entering Stop 3 mode.

0: No effect on HSI16 oscillator

1: HSI16 oscillator forced ON even in Stop mode

Bit 11 **HSION**: HSI16 clock enable

Set and cleared by software. Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby, or Shutdown mode.

Set by hardware to force the HSI16 oscillator ON when STOPWUCK = 1 when leaving Stop modes, or in case of failure of the HSE crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

Bit 10 **MSIPLL0RDY**: MSIRC0 PLL mode ready flag

Set by hardware to indicate that the MSIRC0 PLL mode is locked, when the PLL mode is enabled (MSIPLL0EN = 1). This bit is cleared when the PLL mode is disabled (MSIPLL0EN = 0) or when the MSI is disabled (MSISON = MSIKON = 0).

0: MSIRC0 PLL mode is not ready

1: MSIRC0 PLL mode is ready

Bit 9 **MSIPLL1RDY**: MSIRC1 PLL mode ready flag

Set by hardware to indicate that the MSIRC1 PLL mode is locked, when the PLL mode is enabled (MSIPLL1EN = 1). This bit is cleared when the PLL mode is disabled (MSIPLL1EN = 0) or when the MSI is disabled (MSISON = MSIKON = 0).

0: MSIRC1 PLL mode is not ready

1: MSIRC1 PLL mode is ready

Bit 8 **MSIPLL0FAST**: MSIRC0 PLL mode fast startup

This bit is used only if PLL mode is selected (MSIPLL0EN = 1).

The fast startup feature is not active the first time the PLL mode is selected. The fast startup is active when the MSIRC0 in PLL mode returns from switch off.

0: MSIRC0 PLL normal startup

1: MSIRC0 PLL fast startup

Bit 7 **MSIPLL1FAST**: MSIRC1 PLL mode fast startup

This bit is used only if PLL mode is selected (MSIPLL1EN = 1).

The fast startup feature is not active the first time the PLL mode is selected. The fast startup is active when the MSIRC1 in PLL mode returns from switch off.

0: MSIRC1 PLL normal startup

1: MSIRC1 PLL fast startup

Bit 6 **MSIPLL0EN**: MSIRC0 PLL mode enable

Either MSIS or MSIK, with MSIRC0 selected, must be enabled and ready before enabling MSIRC0 PLL mode.

When MSIPLL0SEL = 0, MSIPLL0EN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware). A hardware protection prevents from enabling MSIPLL0EN if LSE is not ready. This bit is cleared by hardware when LSE is disabled (LSEON = 0), or when the CSS on LSE detects a LSE failure (see RCC_CR).

When MSIPLL0SEL = 1, MSIPLL0EN must be enabled after HSE is enabled (HSEON enabled) and ready (HSERDY set by hardware). A hardware protection prevents from enabling MSIPLL0EN if HSE is not ready, and MSIPLL0SEL is ignored. This bit is cleared by hardware when HSE is disabled (HSEON = 0) or when the CSS on HSE detects a HSE failure.

0: MSIRC0 PLL mode disabled

1: MSIRC0 PLL mode enabled

Bit 5 **MSIPLL1EN**: MSIRC1 PLL mode enable

Either MSIS or MSIK, with MSIRC1 selected, must be enabled and ready before enabling MSIRC1 PLL mode.

When MSIPLL1SEL = 0, MSIPLL1EN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware). A hardware protection prevents from enabling MSIPLL1EN if LSE is not ready. This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the CSS on LSE detects a LSE failure (see RCC_CSR).

When MSIPLL1SEL = 1, MSIPLL1EN must be enabled after HSE is enabled (HSEON enabled) and ready (HSERDY set by hardware). A hardware protection prevents from enabling MSIPLL1EN if HSE is not ready, and MSIPLL1SEL is ignored. This bit is cleared by hardware when HSE is disabled (HSEON = 0) or when the CSS on HSE detects a HSE failure.

0: MSIRC1 PLL mode disabled

1: MSIRC1 PLL mode enabled

Bit 4 **MSIKRDY**: MSIK clock ready flag

Set by hardware to indicate that the MSIK is stable.

0: MSIK (MSI kernel) oscillator not ready

1: MSIK (MSI kernel) oscillator ready

Note: Once the MSIKON bit is cleared, MSIKRDY goes low after six MSIK oscillator clock cycles.

Bit 3 **MSIKON**: MSIK clock enable

Set and cleared by software. Cleared by hardware to stop the MSIK when entering Stop, Standby, or Shutdown mode. Set by hardware to force the MSIK oscillator ON when exiting Standby or Shutdown mode. Set by hardware to force the MSIK oscillator ON when STOPWUCK = 0 or STOPKERWUCK = 0 when exiting Stop modes, or in case of a failure of the HSE oscillator.

0: MSIK (MSI kernel) oscillator disabled

1: MSIK (MSI kernel) oscillator enabled

Bit 2 **MSISRDY**: MSIS clock ready flag

Set by hardware to indicate that the MSIS oscillator is stable after MSISON is set.

0: MSIS (MSI system) oscillator not ready

1: MSIS (MSI system) oscillator ready

Note: Once the MSISON bit is cleared, MSISRDY goes low after six MSIS clock cycles.

Bit 1 **MSIKERON:** MSI enable for some peripheral kernels

This bit is used to force MSI ON even in Stop mode. Keeping the MSI ON in Stop mode allows the communication speed not to be reduced by the MSI startup time. This bit has no effect on MSISON and MSIKON values (see *Peripherals clock gating and autonomous mode* for more details). MSIKERON must be configured at 0 before entering Stop 3 mode.

0: No effect on MSI oscillator

1: MSI oscillator forced ON even in Stop mode.

Bit 0 **MSISON**: MSIS clock enable

Set and cleared by software. Cleared by hardware to stop the MSIS oscillator when entering Stop, Standby, or Shutdown mode. Set by hardware to force the MSIS oscillator ON when exiting Standby or Shutdown mode. Set by hardware to force the MSIS oscillator ON when STOPWUCK = 0 when exiting Stop modes, or in case of a failure of the HSE oscillator.

Set by hardware when used directly or indirectly as system clock.

0: MSIS (MSI system) oscillator OFF

1: MSIS (MSI system) oscillator ON

10.5.2 RCC internal clock source calibration register 1 (RCC_ICSCR1)

Address offset: 0x008

Reset value: 0xB400 0XXX

X is factory-programmed.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSISSEL	MSISDIV[1:0]		MSIKSEL	MSIKDIV[1:0]		MSIPLL1N[1:0]		MSIRGSEL	MSIBIAS	MSIPLL0SEL	MSIPLL1SEL	MSIHSINDIV	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MSICAL0[5:0]						MSICAL1[5:0]					
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **MSISSEL**: MSIS clock source selection

These bits are configured by software to select the RC oscillator used to generate MSIS clock when MSIRGSEL is set.

0: MSIRC0 (96 MHz) is selected as source to generate MSIS.

1: MSIRC1 (24 MHz) is selected as source to generate MSIS.

Note: MSISSEL can be modified when MSIS is OFF (MSISON = 0) or when MSIS is ready (MSISRDY = 1). MSISSEL must not be modified when MSIS is ON and NOT ready (MSISON = 1 and MSISRDY = 0). MSISSEL is kept when the device wakes up from Stop mode.

Bits 30:29 **MSISDIV[1:0]**: MSIS oscillator division

These bits are configured by software to choose the frequency of MSIS oscillator when MSIRGSEL is set.

Condition: MSISSEL bit in RCC_ICSCR1 = 0

00: MSIRC0/1 is selected for MSIS (range 0 around 96 MHz).

01: MSIRC0/2 is selected for MSIS (range 1 around 48 MHz).

10: MSIRC0/4 is selected for MSIS (range 2 around 24 MHz).

11: MSIRC0/8 is selected for MSIS (range 3 around 12 MHz).

Condition: MSISSEL bit in RCC_ICSCR1 = 1

00: MSIRC1/1 is selected for MSIS (range 4 around 24 MHz).

01: MSIRC1/2 is selected for MSIS (range 5 around 12 MHz).

10: MSIRC1/4 is selected for MSIS (range 6 around 6 MHz).

11: MSIRC1/8 is selected for MSIS (range 7 around 3 MHz).

Note: MSISDIV can be modified when MSIS is OFF (MSISON = 0) or when MSIS is ready (MSISRDY = 1). MSISDIV must NOT be modified when MSIS is ON and NOT ready (MSISON = 1 and MSISRDY = 0). MSISDIV is kept when the device wakes up from Stop mode, except when the MSIS range is above 48 MHz. In this case, MSISDIV is changed by hardware into range 1 (48 MHz).The frequencies slightly differ in PLL mode (refer to datasheet for more information).

Bit 28 **MSIKSEL**: MSIK clock source selection

These bits are configured by software to select the RC oscillator used to generate MSIK clock when MSIRGSEL is set.

0: MSIRC0 (96 MHz) is selected as source to generate MSIK.

1: MSIRC1 (24 MHz) is selected as source to generate MSIK.

Note: MSIKSEL can be modified when MSIK is OFF (MSIKON = 0) or when MSIK is ready (MSIKRDY = 1). MSIKSEL must not be modified when MSIK is ON and NOT ready (MSIKON = 1 and MSIKRDY = 0). MSIKSEL is kept when the device wakes up from Stop mode.

Bits 27:26 **MSIKDIV[1:0]**: MSIK oscillator division

These bits are configured by software to choose the frequency of MSIK oscillator when MSIRGSEL is set.

Condition: MSIKSEL bit in RCC_ICSCR1 = 0

00: MSIRC0/1 is selected for MSIK (range 0 around 96 MHz).

01: MSIRC0/2 is selected for MSIK (range 1 around 48 MHz).

10: MSIRC0/4 is selected for MSIK (range 2 around 24 MHz).

11: MSIRC0/8 is selected for MSIK (range 3 around 12 MHz).

Condition: MSIKSEL bit in RCC_ICSCR1 = 1

00: MSIRC1/1 is selected for MSIK (range 4 around 24 MHz).

01: MSIRC1/2 is selected for MSIK (range 5 around 12 MHz).

10: MSIRC1/4 is selected for MSIK (range 6 around 6 MHz.)

11: MSIRC1/8 is selected for MSIK (range 7 around 3 MHz).

Note: MSIKDIV can be modified when MSIK is OFF (MSIKON = 0) or when MSIK is ready (MSIKRDY = 1). MSIKDIV must NOT be modified when MSIK is ON and NOT ready (MSIKON = 1 and MSIKRDY = 0). MSIKDIV is kept when the device wakes up from Stop mode, except when the MSIK range is above 48 MHz. In this case MSIKDIV is changed by hardware into range 1 (48 MHz). The frequencies slightly differ in PLL mode (refer to datasheet for more information).

Bits 25:24 **MSIPLL1N[1:0]**: MSIRC1 PLL mode with LSE multiplication factor

These bits select the frequency of MSIRC1 in PLL mode, impacting range 4, 5, 6 and 7 frequencies. These bits have no effect when the PLL mode is disabled.

Note: Refer to Table 95: MSIS and MSIK ranges per internal MSIRCs (PLL mode disabled) for MSIS/MSIK clock output depends on input and configuration.

Bit 23 **MSIRGSEL**: MSI clock range selection

Set by software to select the MSIS/MSIK frequency range, using MSISSEL/MSIKSEL and MSISDIV[1:0]/MSIKDIV[1:0] bitfields. Writing 0 has no effect.

After exiting Standby or Shutdown mode, or after a reset, this bit is at 0, and the MSIS and MSIK ranges are provided by MSISDIVS[1:0] and MSIKDIVS[1:0] in RCC_CSR.

0: MSIS/MSIK ranges provided by MSISDIVS[1:0] and MSIKDIVS[1:0] in RCC_CSR

1: MSIS/MSIK ranges provided by MSISDIV[1:0] and MSIKDIV[1:0] in RCC_ICSCR1

Bit 22 **MSIBIAS**: MSI bias mode selection

By default, the MSI bias (for both MSIS and MSIK) is in continuous mode in order to maintain the output clocks accuracy. Setting this bit reduces the MSI consumption when the device is in Stop 1 or Stop 2 mode, or when the regulator is in range 2, but it decreases the MSI accuracy

0: MSI bias continuous mode (clock accuracy fast settling time)

1: MSI bias sampling mode when the device is in Stop 1 or Stop 2 mode, or when the regulator is in range 2

Bit 21 **MSIPLL0SEL**: MSIRC0 PLL mode input clock selection

LSE or HSE must be enabled and ready before selecting this oscillator as MSIRC0 input clock.

0: LSE is used as MSIRC0 input clock when PLL mode is selected.

1: HSE or HSE/2 is used as MSIRC0 input clock when PLL mode is selected.

Bit 20 **MSIPLL1SEL**: MSIRC1 PLL mode input clock selection

LSE or HSE must be enabled and ready before selecting this oscillator as MSIRC1 input clock.

0: LSE is used as MSIRC1 input clock when PLL mode is selected.

1: HSE or HSE/2 is used as MSIRC1 input clock when PLL mode is selected.

Bit 19 **MSIHSINDIV**: MSIRCx (x = 0, 1) PLL mode HSE input division

This bit is used to divide the HSE input clock when used by MSIRCx (x = 0, 1) in PLL mode. MSIRCx high-speed input clock must be 16 MHz.

0: HSE (16 MHz) is used as MSI input clock when PLL mode with high-speed clock is selected.

1: HSE (32 MHz)/2 is used as MSI input clock when PLL mode with high-speed clock is selected.

Bits 18:12 Reserved, must be kept at reset value.

Bits 11:6 **MSICAL0[5:0]**: MSIRC0 clock calibration for MSI ranges 0 to 3

These bits are initialized at startup with the factory-programmed MSIRC0 calibration trim value for ranges 0 to 3. When MSITRIM0 is written, MSICAL0 is updated with the sum of MSITRIM0[5:0] and the factory-programmed calibration trim value MSIRC0[5:0].

There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

Bits 5:0 **MSICAL1[5:0]**: MSIRC1 clock calibration for MSI ranges 4 to 7

These bits are initialized at startup with the factory-programmed MSIRC1 calibration trim value for ranges 4 to 7. When MSITRIM1 is written, MSICAL1 is updated with the sum of MSITRIM1[5:0] and the factory-programmed calibration trim value MSIRC1[5:0].

There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

10.5.3 RCC internal clock source calibration register 2 (RCC_ICSCR2)

Address offset: 0x00C

Reset value: 0x0000 0820

Access: no wait state; word, half-word, and byte access

Note: The hardware auto calibration (PLL mode) must not be used in conjunction with software calibration

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MSITRIM0[5:0]						MSITRIM1[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:6 **MSITRIM0[5:0]**: MSIRC0 clock trimming for ranges 0 to 3

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC0[5:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

Bits 5:0 **MSITRIM1[5:0]**: MSIRC1 clock trimming for ranges 4 to 7

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC1[5:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

10.5.4 RCC internal clock source calibration register 3 (RCC_ICSCR3)

Address offset: 0x010

Reset value: 0x0010 0XXX

X is factory-programmed.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **HSITRIM[4:0]:** HSI clock trimming

These bits provide an additional user-programmable trimming value. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the HSI.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

These bits are initialized at startup with the factory-programmed HSI calibration trim value.

10.5.5 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x014

Reset value: 0x0000 0XXX

X is factory-programmed.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **HSI48CAL[8:0]**: HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value.

10.5.6 RCC clock configuration register 1 (RCC_CFGR1)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2; word, half-word, and byte access

1 or 2 wait states are inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	MCO2PRE[2:0]			MCO2SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPKERWUCK	STOPWUCK	SWS[1:0]		SW[1:0]	
										rw	rw	r	r	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]**: Microcontroller clock output prescaler

This prescaler must be configured before MCO output is enabled.

000: MCO divided by 1
001: MCO divided by 2
010: MCO divided by 4
011: MCO divided by 8
100: MCO divided by 16
101: MCO divided by 32
110: MCO divided by 64
111: MCO divided by 128

Bits 27:24 **MCOSEL[3:0]**: Microcontroller clock output

Set and cleared by software.

0000: MCO output disabled, no clock on MCO
0001: SYSCLK system clock selected
0010: MSIS clock selected
0011: HSI16 clock selected
0100: HSE clock selected
0101: LSI clock selected
0110: LSE clock selected
0111: HSI48 clock selected
1000: MSIK clock selected
Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **MCO2PRE[2:0]**: Microcontroller clock output 2 prescaler
This prescaler must be configured before MCO2 output is enabled.
000: MCO2 divided by 1
001: MCO2 divided by 2
010: MCO2 divided by 4
011: MCO2 divided by 8
100: MCO2 divided by 16
101: MCO2 divided by 32
110: MCO2 divided by 64
111: MCO2 divided by 128

Bits 19:16 **MCO2SEL[3:0]**: Microcontroller clock output 2
0000: MCO2 output disabled, no clock on MCO2
0001: SYSCLK system clock selected
0010: MSIS clock selected
0011: HSI16 clock selected
0100: HSE clock selected
0101: LSI clock selected
0110: LSE clock selected
0111: HSI48 clock selected
1000: MSIK clock selected
Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **STOPKERWUCK**: Wake-up from Stop kernel clock automatic enable selection
This bit is used to enable automatically another oscillator than the one used as wake-up from Stop system clock. This helps to save its wake-up time when this oscillator is needed as peripheral independent kernel clock.
0: MSIK oscillator automatically enabled when exiting Stop mode or when a CSS on HSE event occurs
1: HSI16 oscillator automatically enabled when exiting Stop mode or when a CSS on HSE event occurs

Note: STOPKERWUCK must be programmed with the same value than STOPWUCK to avoid the other oscillator power-on after CSS event or Stop mode exit.

Bit 4 **STOPWUCK**: Wake-up from Stop and CSS backup clock selection
This bit is used to select the system clock used when exiting Stop mode. The selected clock is also used as emergency clock for the clock security system on HSE.
0: MSIS oscillator selected as wake-up from stop clock and CSS backup clock
1: HSI16 oscillator selected as wake-up from stop clock and CSS backup clock

Note: STOPWUCK must not be modified when the CSS is enabled by HSECSSON bit in RCC_CR and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW = 10). STOPKERWUCK must be programmed with the same value than STOPWUCK to avoid the other oscillator power-on after CSS event or Stop mode exit.

Bits 3:2 **SWS[1:0]**: System clock switch status
Set and cleared by hardware to indicate which clock source is used as system clock.
00: MSIS oscillator used as system clock
01: HSI16 oscillator used as system clock
10: HSE used as system clock
11: Reserved

Bits 1:0 **SW[1:0]**: System clock switch

These bits are used to select system clock source (SYSCLK). Configured by hardware to force MSIS oscillator selection when exiting Standby or Shutdown mode. Configured by hardware to force MSIS or HSI16 oscillator selection when exiting Stop mode, or in case of HSE oscillator failure, depending on STOPWUCK value.

00: MSIS selected as system clock

01: HSI16 selected as system clock

10: HSE selected as system clock

11: Reserved

10.5.7 RCC clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

Access: word, half-word, and byte access

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB2 prescaler

These bits are used to configure the division factor of the APB2 clock (PCLK2).

000-011: PCLK2 = HCLK not divided

100: PCLK2 = HCLK divided by 2

101: PCLK2 = HCLK divided by 4

110: PCLK2 = HCLK divided by 8

111: PCLK2 = HCLK divided by 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]**: APB1 prescaler

These bits are used to configure the division factor of the APB1 clock (PCLK1).

000-011: PCLK1 = HCLK not divided

100: PCLK1 = HCLK divided by 2

101: PCLK1 = HCLK divided by 4

110: PCLK1 = HCLK divided by 8

111: PCLK1 = HCLK divided by 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler

These bits are used to configure the division factor of the AHB clock (HCLK). Depending on the device voltage range, the software must set these bits correctly to ensure that the system frequency does not exceed the maximum allowed frequency (for more details, refer to *Table 97: Bus maximum frequency*). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value is taken into account.

0000-0111: HCLK = SYSCLK not divided
1000: HCLK = SYSCLK divided by 2
1001: HCLK = SYSCLK divided by 4
1010: HCLK = SYSCLK divided by 8
1011: HCLK = SYSCLK divided by 16
1100: HCLK = SYSCLK divided by 64
1101: HCLK = SYSCLK divided by 128
1110: HCLK = SYSCLK divided by 256
1111: HCLK = SYSCLK divided by 512

10.5.8 RCC clock configuration register 3 (RCC_CFGR3)

Address offset: 0x024

Reset value: 0x0000 0000

Access: word, half-word, and byte access

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE3[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE3[2:0]**: APB3 prescaler

These bits are used to configure the division factor of the APB3 clock (PCLK3).

000-011: PCLK3 = HCLK not divided
100: PCLK3 = HCLK divided by 2
101: PCLK3 = HCLK divided by 4
110: PCLK3 = HCLK divided by 8
111: PCLK3 = HCLK divided by 16

Bits 3:0 Reserved, must be kept at reset value.

10.5.9 RCC clock configuration register 4 (RCC_CFGR4)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOSTDIV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTSEL[1:0]	
rw	rw	rw	rw											rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **BOOSTDIV[3:0]**: EPOD booster input clock prescaler

These bits are used to configure the division factor for the EPOD booster input clock, when it is HSI16 or HSE. The division factor is forced by hardware when the EPOD booster input clock is MSIS. In this case, these bits are read only. The EPOD booster input frequency is BOOSTSEL oscillator/BOOSTDIV, and must be between 3 and 16 MHz (see *Section 10.1: RCC introduction*).

0000: Divided by 1 (bypass)
0001: Divided by 2
0010: Divided by 4
0011: Divided by 6
0100: Divided by 8
0101: Divided by 10
0110: Divided by 12
0111: Divided by 14
1000: Divided by 16
Others: Reserved

Bits 11:2 Reserved, must be kept at reset value.

Bits 1:0 **BOOSTSEL[1:0]**: EPOD booster input clock source selection

These bits are used to selected to EPOD booster input clock source oscillator. The booster clock source must be enabled and configured before enabling the booster. BOOSTSEL must be kept at 00 to save power when the booster is disabled.

00: No clock
01: MSIS selected as EPOD booster clock source
10: HSI16 selected as EPOD booster clock source
11: HSE selected as EPOD booster clock source

10.5.10 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x050

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LSECSSIE	MSIKRDYIE	Res.	MSIPLLHSUIE	MSIPLLUIE	MSIPLL0RDYIE	MSIPLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	MSISRDYIE	LSERDYIE	LSIRDYIE
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSIE**: LSE clock security system interrupt enable
0: LSE CSS interrupt disabled
1: LSE CSS interrupt enabled

Bit 11 **MSIKRDYIE**: MSIK ready interrupt enable
0: MSIK ready interrupt disabled
1: MSIK ready interrupt enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MSIPLLHSUIE**: MSI PLL mode with HSE unlock interrupt enable
0: MSI PLL mode with HSE unlock interrupt disabled
1: MSI PLL mode with HSE unlock interrupt enabled

Bit 8 **MSIPLLUIE**: MSI PLL mode with LSE unlock interrupt enable
0: MSI PLL mode with LSE unlock interrupt disabled
1: MSI PLL mode with LSE unlock interrupt enabled

Bit 7 **MSIPLL0RDYIE**: MSIRC0 PLL mode ready interrupt enable
0: MSIRC0 PLL mode lock interrupt disabled
1: MSIRC0 PLL mode lock interrupt enabled

Bit 6 **MSIPLL1RDYIE**: MSIRC1 PLL mode ready interrupt enable
0: MSIRC1 PLL mode lock interrupt disabled
1: MSIRC1 PLL mode lock interrupt enabled

Bit 5 **HSI48RDYIE**: HSI48 ready interrupt enable
0: HSI48 ready interrupt disabled
1: HSI48 ready interrupt enabled

Bit 4 **HSERDYIE**: HSE ready interrupt enable
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable
0: HSI16 ready interrupt disabled
1: HSI16 ready interrupt enabled

Bit 2 **MSISRDYIE**: MSIS ready interrupt enable
0: MSIS ready interrupt disabled
1: MSIS ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

10.5.11 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x054

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LSECSSF	MSIKRDYF	CSSF	MSIPLLHSUF	MSIPLLUF	MSIPLL0RDYF	MSIPLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	MSISRDYF	LSERDYF	LSIRDYF
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSF**: LSE clock security system interrupt flag
Set by hardware when a failure is detected on LSE and LSECSSIE is set. Cleared by software setting the LSECSSC bit.
0: No LSE clock security interrupt
1: LSE clock security interrupt

Bit 11 **MSIKRDYF**: MSIK ready interrupt flag
Set by hardware when the MSIK clock becomes stable and MSIKRDYIE is set. Cleared by software setting the MSIKRDYC bit.
0: No MSIK oscillator clock ready interrupt
1: MSIK oscillator clock ready interrupt

Bit 10 **CSSF**: Clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator. Cleared by software setting the CSSC bit.
0: No HSE clock security interrupt
1: HSE clock security interrupt

Bit 9 **MSIPLLHSUF**: MSI PLL mode with HSE unlock interrupt flag

Set by hardware when the PLL mode of MSI using HSE input is unlocked due to HSE failure and MSIPLLHSUIE is set. This occurs when MSI PLL mode using HSE is ready and HSE clock is stopped. In this case, the MSI automatically exits the PLL mode, and MSIPLL0EN/MSIPLL1EN must be cleared by software.

Cleared by software setting the MSIPLLHSUC bit.

0: No MSI PLL mode with HSE unlock interrupt

1: MSI PLL mode with HSE unlock interrupt

Bit 8 **MSIPLLUF**: MSI PLL mode with LSE unlock interrupt flag

Set by hardware when the PLL mode of MSI using LSE input is unlocked due to LSE failure and MSIPLLUIE is set. This occurs when MSI PLL mode using LSE is ready and LSE clock is stopped. In this case, the MSI automatically exits the PLL mode and MSIPLL0EN/MSIPLL1EN must be cleared by software.

Cleared by software setting the MSIPLLUC bit.

0: No MSI PLL mode with LSE unlock interrupt

1: MSI PLL mode with LSE unlock interrupt

Bit 7 **MSIPLL0RDYF**: MSIRC0 PLL mode ready interrupt flag

Set by hardware when the MSIRC0 clock in PLL mode becomes stable and MSIPLL0RDYIE is set. Cleared by software setting the MSIPLL0RDYC bit.

0: No MSIRC0 PLL mode ready interrupt

1: MSIRC0 PLL mode ready interrupt

Bit 6 **MSIPLL1RDYF**: MSIRC1 PLL mode ready interrupt enable

Set by hardware when the MSIRC1 clock in PLL mode becomes stable and MSIPLL1RDYIE is set. Cleared by software setting the MSIPLL1RDYC bit.

0: No MSIRC1 PLL mode ready interrupt

1: MSIRC1 PLL mode ready interrupt

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag

Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set. Cleared by software setting the HSI48RDYC bit.

0: No HSI48 ready interrupt

1: HSI48 ready interrupt

Bit 4 **HSERDYF**: HSE ready interrupt flag

Set by hardware when the HSE clock becomes stable and HSERDYIE is set. Cleared by software setting the HSERDYC bit.

0: No HSE ready interrupt

1: HSE ready interrupt

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag

Set by hardware when the HSI16 clock becomes stable and HSIRDYIE is set in a response to setting the HSION (see RCC_CR). When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated. Cleared by software setting the HSIRDYC bit.

0: No HSI16 ready interrupt

1: HSI16 ready interrupt

Bit 2 **MSISRDYF**: MSIS ready interrupt flag

Set by hardware when the MSIS clock becomes stable and MSISRDYIE is set. Cleared by software setting the MSISRDYC bit.

0: No MSIS ready interrupt

1: MSIS ready interrupt

Bit 1 **LSERDYF**: LSE ready interrupt flag
Set by hardware when the LSE clock becomes stable and LSERDYIE is set. Cleared by software setting the LSERDYC bit.
0: No LSE ready interrupt
1: LSE ready interrupt

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set. Cleared by software setting the LSIRDYC bit.
0: No LSI ready interrupt
1: LSI ready interrupt

10.5.12 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x058

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LSECSSC	MSIKRDYC	CSSC	MSIPLLHSUC	MSIPLLUC	MSIPLL0RDYC	MSIPLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	MSISRDYC	LSERDYC	LSIRDYC
			w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSC**: LSE CSS interrupt clear
Writing this bit to 1 clears the LSECSSF flag. Writing 0 has no effect.

Bit 11 **MSIKRDYC**: MSIK oscillator ready interrupt clear
Writing this bit to 1 clears the MSIKRDYF flag. Writing 0 has no effect.

Bit 10 **CSSC**: Clock security system interrupt clear
Writing this bit to 1 clears the CSSF flag. Writing 0 has no effect.

Bit 9 **MSIPLLHSUC**: MSI PLL mode with HSE unlock interrupt clear
Writing this bit to 1 clears the MSIPLLHSUF flag. Writing 0 has no effect.

Bit 8 **MSIPLLUC**: MSI PLL mode with LSE unlock interrupt clear
Writing this bit to 1 clears the MSIPLLLSUF flag. Writing 0 has no effect.

Bit 7 **MSIPLL0RDYC**: MSIRC0 PLL mode ready interrupt clear
Writing this bit to 1 clears the MSIPLLHSUF flag. Writing 0 has no effect.

Bit 6 **MSIPLL1RDYC**: MSIRC1 PLL mode ready interrupt clear
Writing this bit to 1 clears the MSIPLLLSUF flag. Writing 0 has no effect.

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Writing this bit to 1 clears the HSI48RDYF flag. Writing 0 has no effect.

Bit 4 **HSERDYC**: HSE ready interrupt clear
Writing this bit to 1 clears the HSERDYF flag. Writing 0 has no effect.

Bit 3 **HSIRDYC**: HSI16 ready interrupt clear
Writing this bit to 1 clears the HSIRDYF flag. Writing 0 has no effect.

Bit 2 **MSISRDYC**: MSIS ready interrupt clear
Writing this bit to 1 clears the MSISRDYF flag. Writing 0 has no effect.

Bit 1 **LSERDYC**: LSE ready interrupt clear
Writing this bit to 1 clears the LSERDYF flag. Writing 0 has no effect.

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Writing this bit to 1 clears the LSIRDYF flag. Writing 0 has no effect.

10.5.13 RCC AHB1 peripheral reset register 1 (RCC_AHB1RSTR1)

Address offset: 0x060

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGRST	TSCRST
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RST	Res.	Res.	GPDMA1RST
			rw									rw			rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGRST**: RAMCFG reset
0: No effect
1: Reset RAMCFG

Bit 16 **TSCRST**: TSC reset
0: No effect
1: Reset TSC

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset
0: No effect
1: Reset CRC

Bits 11:4 Reserved, must be kept at reset value.

Bit 3 **ADF1RST**: ADF1 reset
0: No effect
1: Reset ADF1

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **GPDMA1RST**: GPDMA1 reset
0: No effect
1: Reset GPDMA1

10.5.14 RCC AHB2 peripheral reset register 1 (RCC_AHB2RSTR1)

Address offset: 0x064

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SDMMC1RST	Res.	Res.	Res.	Res.	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST
				rw						rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1RST	ADC12RST	Res.	Res.	GPIOHRST	GPIOGRST	Res.	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
				rw	rw			rw	rw		rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SDMMC1RST**: SDMMC1 reset
0: No effect
1: Reset SDMMC1

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **CCBRST**: CCB reset
0: No effect
1: Reset CCB
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bit 20 **SAESRST**: SAES hardware accelerator reset
0: No effect
1: Reset SAES
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bit 19 **PKARST**: PKA reset
0: No effect
1: Reset PKA
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bit 18 **RNGRST**: Random number generator reset
0: No effect
1: Reset RNG
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bit 17 **HASHRST**: HASH reset
0: No effect
1: Reset HASH
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bit 16 **AESRST**: AES hardware accelerator reset
0: No effect
1: Reset AES
Note: All the peripheral clocks of the CCB subsystem must be disabled before setting this bit.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1RST**: DAC1 reset
0: No effect
1: Reset DAC1

Bit 10 **ADC12RST**: ADC12 reset
0: No effect
1: Reset ADC12

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **GPIOiRST**: I/O port i reset (i = H to G)
0: No effect
1: Reset I/O port i

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **GPIOiRST**: I/O port i reset (i = E to A)
0: No effect
1: Reset I/O port i

10.5.15 RCC AHB2 peripheral reset register 2 (RCC_AHB2RSTR2)

Address offset: 0x068

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1RST	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1RST**: OCTOSPI1 reset
0: No effect
1: Reset OCTOSPI1

Bits 3:0 Reserved, must be kept at reset value.

10.5.16 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x074

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	VREFRST	OPAMPRST	Res.	Res.	Res.	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	Res.	Res.
		rw	rw				rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2RST	Res.	Res.	Res.	Res.	Res.	SPI3RST	Res.	Res.	TIM7RST	TIM6RST	Res.	TIM4RST	TIM3RST	TIM2RST
	rw						rw			rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **VREFRST**: VREFBUF reset
- 0: No effect
- 1: Reset VREFBUF

Bit 28 **OPAMPRST**: OPAMP reset
- 0: No effect
- 1: Reset OPAMP

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **CRSRST**: CRS reset
- 0: No effect
- 1: Reset the CRS

Bit 23 **I3C1RST**: I3C1 reset
- 0: No effect
- 1: Reset the I3C1

Bit 22 **I2C2RST**: I2C2 reset
- 0: No effect
- 1: Reset the I2C2

Bit 21 **I2C1RST**: I2C1 reset
- 0: No effect
- 1: Reset the I2C1

Bit 20 **UART5RST**: UART5 reset
- 0: No effect
- 1: Reset UART5

Bit 19 **UART4RST**: UART4 reset
- 0: No effect
- 1: Reset UART4

Bit 18 **USART3RST**: USART3 reset
- 0: No effect
- 1: Reset USART3

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SPI2RST**: SPI2 reset
0: No effect
1: Reset SPI2

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **SPI3RST**: SPI3 reset
0: No effect
1: Reset SPI3

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TIMjRST**: TIMj reset (j = 7 to 6)
0: No effect
1: Reset TIMj

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TIMjRST**: TIMj reset (j = 4 to 2)
0: No effect
1: Reset TIMj

10.5.17 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x078

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1RST	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	Res.
						rw				rw					

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1RST**: FDCAN1 reset
0: No effect
1: Reset FDCAN1

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST**: LPTIM2 reset
0: No effect
1: Reset LPTIM2

Bits 4:0 Reserved, must be kept at reset value.

10.5.18 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2RST	Res.	Res.	USB1RST	Res.	Res.	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
				rw			rw			rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **I3C2RST**: I3C2 reset
0: No effect
1: Reset I3C2

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **USB1RST**: USB1 reset
0: No effect
1: Reset USB1

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1RST**: SAI1 reset
0: No effect
1: Reset SAI1

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **TIMiRST**: TIMi reset (i = 17 to 15)
0: No effect
1: Reset TIMi

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 reset
0: No effect
1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST**: SPI1 reset
0: No effect
1: Reset SPI1

Bit 11 **TIM1RST**: TIM1 reset
0: No effect
1: Reset TIM1

Bits 10:0 Reserved, must be kept at reset value.

10.5.19 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPRST	Res.	LPTIM4RST	LPTIM3RST	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	Res.	Res.	Res.	Res.	SYSCFGRST	Res.
rw		rw	rw	rw				rw	rw					rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **COMPRST**: COMP reset
0: No effect
1: Reset COMP

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LPTIMiRST**: LPTIMi reset (i = 4 to 3)
0: No effect
1: Reset LPTIMi

Bit 11 **LPTIM1RST**: LPTIM1 reset
0: No effect
1: Reset LPTIM1

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3RST**: I2C3 reset
0: No effect
1: Reset I2C3

Bit 6 **LPUART1RST**: LPUART1 reset
0: No effect
1: Reset LPUART1

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGRST**: SYSCFG reset
0: No effect
1: Reset SYSCFG

Bit 0 Reserved, must be kept at reset value.

10.5.20 RCC AHB1 peripheral clock enable register 1 (RCC_AHB1ENR1)

Address offset: 0x088

Reset value: 0x8000 0100

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	TSCEN
rw							rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	ADF1EN	Res.	Res.	GPDMA1EN
			rw				rw					rw			rw

Bit 31 **SRAM1EN**: SRAM1 clock enable
0: SRAM1 clock disabled
1: SRAM1 clock enabled

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1EN**: GTZC1 clock enable
0: GTZC1 clock disabled
1: GTZC1 clock enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG clock enable
0: RAMCFG clock disabled
1: RAMCFG clock enabled

Bit 16 **TSCEN**: Touch sensing controller clock enable
0: TSC clock disabled
1: TSC clock enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
0: CRC clock disabled
1: CRC clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: FLASH clock enable
This bit can be disabled only when the flash memory is in power-down mode.
0: FLASH clock disabled
1: FLASH clock enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ADF1EN**: ADF1 clock enable
0: ADF1 clock disabled
1: ADF1 clock enabled

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **GPDMA1EN**: GPDMA1 clock enable
0: GPDMA1 clock disabled
1: GPDMA1 clock enabled

10.5.21 RCC AHB2 peripheral clock enable register 1 (RCC_AHB2ENR1)

Address offset: 0x08C

Reset value: 0x4000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2EN	Res.	Res.	SDMMC1EN	Res.	Res.	Res.	Res.	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
	rw			rw						rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1EN	ADC12EN	Res.	Res.	GPIOHEN	GPIOGEN	Res.	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
				rw	rw			rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2EN**: SRAM2 clock enable
0: SRAM2 clock disabled
1: SRAM2 clock enabled

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SDMMC1EN**: SDMMC1 clock enable
0: SDMMC1 clock disabled
1: SDMMC1 clock enabled

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **CCBEN**: CCB clock enable
0: CCB clock disabled
1: CCB clock enabled

Bit 20 **SAESEN**: SAES clock enable
0: SAES clock disabled
1: SAES clock enabled

Bit 19 **PKAEN**: PKA clock enable
0: PKA clock disabled
1: PKA clock enabled

Bit 18 **RNGEN**: RNG clock enable
0: RNG clock disabled
1: RNG clock enabled

Bit 17 **HASHEN**: HASH clock enable
0: HASH clock disabled
1: HASH clock enabled

Bit 16 **AESEN**: AES clock enable
0: AES clock disabled
1: AES clock enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1EN**: DAC1 clock enable
0: DAC1 clock disabled
1: DAC1 clock enabled

Bit 10 **ADC12EN**: ADC12 clock enable
0: ADC12 clock disabled
1: ADC12 clock enabled

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **GPIOiEN**: I/O port i clock enable (i = H to G)
0: I/O port i clock disabled
1: I/O port i clock enabled

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **GPIOiEN**: I/O port i clock enable (i = E to A)
0: I/O port i clock disabled
1: I/O port i clock enabled

10.5.22 RCC AHB2 peripheral clock enable register 2 (RCC_AHB2ENR2)

Address offset: 0x090

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1EN	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1EN**: OCTOSPI1 clock enable
0: OCTOSPI1 clock disabled
1: OCTOSPI1 clock enabled

Bits 3:0 Reserved, must be kept at reset value.

10.5.23 RCC AHB1 peripheral clock enable register 2 (RCC_AHB1ENR2)

Address offset: 0x094

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWREN	Res.	Res.
													rw		

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **PWREN**: PWR clock enable
0: PWR clock disabled
1: PWR clock enabled

Bits 1:0 Reserved, must be kept at reset value.

10.5.24 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address offset: 0x09C

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RTCAPBEN	VREFEN	OPAMPEN	Res.	Res.	Res.	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	Res.	Res.
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	SPI3EN	Res.	Res.	TIM7EN	TIM6EN	Res.	TIM4EN	TIM3EN	TIM2EN
	rw			rs			rw			rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **RTCAPBEN**: RTC and TAMP APB clock enable
0: RTC and TAMP APB clock disabled
1: RTC and TAMP APB clock enabled

Bit 29 **VREFEN**: VREFBUF clock enable
0: VREFBUF clock disabled
1: VREFBUF clock enabled

Bit 28 **OPAMPEN**: OPAMP clock enable
0: OPAMP clock disabled
1: OPAMP clock enabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **CRSEN**: CRS clock enable
0: CRS clock disabled
1: CRS clock enabled

Bit 23 **I3C1EN**: I3C1 clock enable
0: I3C1 clock disabled
1: I3C1 clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable
0: I2C2 clock disabled
1: I2C2 clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
0: I2C1 clock disabled
1: I2C1 clock enabled

Bit 20 **UART5EN**: UART5 clock enable
0: UART5 clock disabled
1: UART5 clock enabled

Bit 19 **UART4EN**: UART4 clock enable
0: UART4 clock disabled
1: UART4 clock enabled

Bit 18 **USART3EN**: USART3 clock enable
0: USART3 clock disabled
1: USART3 clock enabled

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SPI2EN**: SPI2 clock enable
0: SPI2 clock disabled
1: SPI2 clock enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
Set by software to enable the window watchdog clock. Reset by hardware system reset. This bit can also be set by hardware if the WWDG_SW option bit is reset.
0: WWDG clock disabled
1: WWDG clock enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SPI3EN**: SPI3 clock enable
0: SPI3 clock disabled
1: SPI3 clock enabled

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TIMjEN**: TIMj clock enable (j = 7 to 6)
0: TIMj clock disabled
1: TIMj clock enabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TIMjEN**: TIMj clock enable (j = 4 to 2)
0: TIMj clock disabled
1: TIMj clock enabled

10.5.25 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x0A0

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral register read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1EN	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.
						rw				rw					

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1EN**: FDCAN1 clock enable
0: FDCAN1 clock disabled
1: FDCAN1 clock enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN**: LPTIM2 clock enable
0: LPTIM2 clock disabled
1: LPTIM2 clock enabled

Bits 4:0 Reserved, must be kept at reset value.

10.5.26 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

Access: word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral register read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2EN	Res.	Res.	USB1EN	Res.	Res.	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
				rw			rw			rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **I3C2EN**: I3C2 clock enable
0: I3C2 clock disabled
1: I3C2 clock enabled

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **USB1EN**: USB1 clock enable
0: USB1 clock disabled
1: USB1 clock enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1EN**: SAI1 clock enable
0: SAI1 clock disabled
1: SAI1 clock enabled

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **TIMiEN**: TIMi clock enable (i = 17 to 15)
0: TIMi clock disabled
1: TIMi clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1clock enable
0: USART1 clock disabled
1: USART1 clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable
0: SPI1 clock disabled
1: SPI1 clock enabled

Bit 11 **TIM1EN**: TIM1 clock enable
0: TIM1 clock disabled
1: TIM1 clock enabled

Bits 10:0 Reserved, must be kept at reset value.

10.5.27 RCC APB3 peripheral clock enable register (RCC_APB3ENR)

Address offset: 0x0A8

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

Note: When the peripheral clock is not active, the peripheral register read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPEN	Res.	LPTIM4EN	LPTIM3EN	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	Res.	Res.	Res.	Res.	SYSCFGEN	Res.
rw		rw	rw	rw				rw	rw					rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **COMPEN**: COMP clock enable
0: COMP clock disabled
1: COMP clock enabled

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LPTIMiEN**: LPTIMi clock enable (i = 4 to 3)
0: LPTIMi clock disabled
1: LPTIMi clock enabled

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
0: LPTIM1 clock disabled
1: LPTIM1 clock enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3EN**: I2C3 clock enable
0: I2C3 clock disabled
1: I2C3 clock enabled

Bit 6 **LPUART1EN**: LPUART1 clock enable
0: LPUART1 clock disabled
1: LPUART1 clock enabled

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGEN**: SYSCFG clock enable
0: SYSCFG clock disabled
1: SYSCFG clock enabled

Bit 0 Reserved, must be kept at reset value.

10.5.28 RCC AHB1 peripheral clock enable in Sleep mode register (RCC_AHB1SLPENR1)

Address offset: 0x0B0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1SLPEN	Res.	ICACHESLPEN	Res.	Res.	Res.	Res.	GTZC1SLPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSLPEN	TSCSLPEN
rw		rw					rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCSLPEN	Res.	Res.	Res.	FLASHSLPEN	Res.	Res.	Res.	Res.	ADF1SLPEN	Res.	Res.	GPDMA1SLPEN
			rw				rw					rw			rw

Bit 31 **SRAM1SLPEN**: SRAM1 clock enable during Sleep mode
0: SRAM1 clock disabled by the clock gating during Sleep mode
1: SRAM1 clock enabled by the clock gating during Sleep mode

Bit 30 Reserved, must be kept at reset value.

Bit 29 **ICACHESLPEN**: ICACHE clock enable during Sleep mode
0: ICACHE clock disabled by the clock gating during Sleep mode
1: ICACHE clock enabled by the clock gating during Sleep mode

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1SLPEN**: GTZC1 clock enable during Sleep mode
0: GTZC1 clock disabled by the clock gating during Sleep mode
1: GTZC1 clock enabled by the clock gating during Sleep mode

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGSLPEN**: RAMCFG clock enable during Sleep mode
0: RAMCFG clock disabled by the clock gating during Sleep mode
1: RAMCFG clock enabled by the clock gating during Sleep mode

Bit 16 **TSCSLPEN**: TSC clock enable during Sleep mode
0: TSC clock disabled by the clock gating during Sleep mode
1: TSC clock enabled by the clock gating during Sleep mode

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSLPEN**: CRC clock enable during Sleep mode
0: CRC clock disabled by the clock gating during Sleep mode
1: CRC clock enabled by the clock gating during Sleep mode

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHSLPEN**: FLASH clock enable during Sleep mode
0: FLASH clock disabled by the clock gating during Sleep mode
1: FLASH clock enabled by the clock gating during Sleep mode

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ADF1SLPEN**: ADF1 clock enable during Sleep mode.
0: ADF1 clock disabled by the clock gating during Sleep mode
1: ADF1 clock enabled by the clock gating during Sleep mode

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **GPDMA1SLPEN**: GPDMA1 clock enable during Sleep mode
0: GPDMA1 clock disabled by the clock gating during Sleep mode
1: GPDMA1 clock enabled by the clock gating during Sleep mode

10.5.29 RCC AHB2 peripheral clock enable in Sleep mode register 1 (RCC_AHB2SLPENR1)

Address offset: 0x0B4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one

or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2SLPEN	Res.	Res.	SDMMC1SLPEN	Res.	Res.	Res.	Res.	Res.	CCBSLPEN	SAESSLPEN	PKASLPEN	RNGSLPEN	HASHSLPEN	AESSLPEN
	rw			rw						rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1SLPEN	ADC12SLPEN	Res.	Res.	GPIOHSLPEN	GPIOGSLPEN	Res.	GPIOESLPEN	GPIODSLPEN	GPIOCSLPEN	GPIOBSLPEN	GPIOASLPEN
				rw	rw			rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2SLPEN**: SRAM2 clock enable during Sleep mode
0: SRAM2 clock disabled by the clock gating during Sleep mode
1: SRAM2 clock enabled by the clock gating during Sleep mode

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SDMMC1SLPEN**: SDMMC1 clock enable during Sleep mode
0: SDMMC1 clock disabled by the clock gating during Sleep mode
1: SDMMC1 clock enabled by the clock gating during Sleep mode

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **CCBSLPEN**: CCB accelerator clock enable during Sleep mode
0: CCB clock disabled by the clock gating during Sleep mode
1: CCB clock enabled by the clock gating during Sleep mode

Bit 20 **SAESSLPEN**: SAES accelerator clock enable during Sleep mode
0: SAES clock disabled by the clock gating during Sleep mode
1: SAES clock enabled by the clock gating during Sleep mode

Bit 19 **PKASLPEN**: PKA clock enable during Sleep mode
0: PKA clock disabled by the clock gating during Sleep mode
1: PKA clock enabled by the clock gating during Sleep mode

Bit 18 **RNGSLPEN**: RNG clock enable during Sleep mode
0: RNG clock disabled by the clock gating during Sleep mode
1: RNG clock enabled by the clock gating during Sleep mode

Bit 17 **HASHSLPEN**: HASH clock enable during Sleep mode
0: HASH clock disabled by the clock gating during Sleep mode
1: HASH clock enabled by the clock gating during Sleep mode

Bit 16 **AESSLPEN**: AES clock enable during Sleep mode
0: AES clock disabled by the clock gating during Sleep mode
1: AES clock enabled by the clock gating during Sleep mode

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DAC1SLPEN**: DAC1 clock enable during Sleep mode
0: DAC1 clock disabled by the clock gating during Sleep mode
1: DAC1 clock enabled by the clock gating during Sleep mode

Bit 10 **ADC12SLPEN**: ADC12 clock enable during Sleep mode
0: ADC12 clock disabled by the clock gating during Sleep mode
1: ADC12 clock enabled by the clock gating during Sleep mode

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **GPIOiSLPEN**: I/O port i clock enable during Sleep mode (i = H to G)
0: I/O port i clocks disabled by the clock gating during Sleep mode
1: I/O port i clock enabled by the clock gating during Sleep mode

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **GPIOiSLPEN**: I/O port i clock enable during Sleep mode (i = E to A)
0: I/O port i clocks disabled by the clock gating during Sleep mode
1: I/O port i clock enabled by the clock gating during Sleep mode

10.5.30 RCC AHB2 peripheral clock enable in Sleep mode register 2 (RCC_AHB2SLPENR2)

Address offset: 0x0B8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1SLPEN	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1SLPEN**: OCTOSPI1 clock enable during Sleep mode
0: OCTOSPI1 clock disabled by the clock gating during Sleep mode
1: OCTOSPI1 clock enabled by the clock gating during Sleep mode

Bits 3:0 Reserved, must be kept at reset value.

10.5.31 RCC AHB1 peripheral clock enable in Sleep mode register 2 (RCC_AHB1SLPENR2)

Address offset: 0x0BC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWRSLPEN	Res.	Res.
													rw		

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **PWRSLPEN**: PWR clock enable during Sleep mode
0: PWR clock disabled by the clock gating during Sleep mode
1: PWR clock enabled by the clock gating during Sleep mode

Bits 1:0 Reserved, must be kept at reset value.

10.5.32 RCC APB1 peripheral clock enable in Sleep mode register 1 (RCC_APB1SLPENR1)

Address offset: 0x0C4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RTCAPBSLPEN	VREFSLPEN	OPAMPSLPEN	Res.	Res.	Res.	CRSSLPEN	I3C1SLPEN	I2C2SLPEN	I2C1SLPEN	UART5SLPEN	UART4SLPEN	USART3SLPEN	Res.	Res.
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2SLPEN	Res.	Res.	WWDGSLPEN	Res.	Res.	SPI3SLPEN	Res.	Res.	TIM7SLPEN	TIM6SLPEN	Res.	TIM4SLPEN	TIM3SLPEN	TIM2SLPEN
	rw			rw			rw			rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **RTCAPBSLPEN**: RTC and TAMP APB clock enable during Sleep mode
0: RTC and TAMP APB clock disabled by the clock gating during Sleep mode
1: RTC and TAMP APB clock enabled by the clock gating during Sleep mode

Bit 29 **VREFSLPEN**: VREFBUF clock enable during Sleep mode
0: VREFBUF clock disabled by the clock gating during Sleep mode
1: VREFBUF clock enabled by the clock gating during Sleep mode

Bit 28 **OPAMPSLPEN**: OPAMP clock enable during Sleep mode
0: OPAMP clock disabled by the clock gating during Sleep mode
1: OPAMP clock enabled by the clock gating during Sleep mode

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **CRSSLPEN**: CRS clock enable during Sleep mode
0: CRS clock disabled by the clock gating during Sleep mode
1: CRS clock enabled by the clock gating during Sleep mode

Bit 23 **I3C1SLPEN**: I3C1 clock enable during Sleep mode
0: I3C1 clock disabled by the clock gating during Sleep mode
1: I3C1 clock enabled by the clock gating during Sleep mode

Bit 22 **I2C2SLPEN**: I2C2 clock enable during Sleep mode
0: I2C2 clock disabled by the clock gating during Sleep mode
1: I2C2 clock enabled by the clock gating during Sleep mode

Bit 21 **I2C1SLPEN**: I2C1 clock enable during Sleep mode
0: I2C1 clock disabled by the clock gating during Sleep mode
1: I2C1 clock enabled by the clock gating during Sleep mode

Bit 20 **UART5SLPEN**: UART5 clock enable during Sleep mode
0: UART5 clock disabled by the clock gating during Sleep mode
1: UART5 clock enabled by the clock gating during Sleep mode

Bit 19 **UART4SLPEN**: UART4 clock enable during Sleep mode
0: UART4 clock disabled by the clock gating during Sleep mode
1: UART4 clock enabled by the clock gating during Sleep mode

Bit 18 **USART3SLPEN**: USART3 clock enable during Sleep mode
0: USART3 clock disabled by the clock gating during Sleep mode
1: USART3 clock enabled by the clock gating during Sleep mode

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SPI2SLPEN**: SPI2 clock enable during Sleep mode
0: SPI2 clock disabled by the clock gating during Sleep mode
1: SPI2 clock enabled by the clock gating during Sleep mode

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSLPEN**: WWDG clock enable during Sleep mode
This bit is forced to 1 by hardware when the hardware WWDG option is activated.
0: WWDG clock disabled by the clock gating during Sleep mode
1: WWDG clock enabled by the clock gating during Sleep mode

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SPI3SLPEN**: SPI3 clock enable during Sleep mode
0: SPI3 clock disabled by the clock gating during Sleep mode
1: SPI3 clock enabled by the clock gating during Sleep mode

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TIMjSLPEN**: TIMj clock enable during Sleep mode (j = 7 to 6)
0: TIMj clock disabled by the clock gating during Sleep mode
1: TIMj clock enabled by the clock gating during Sleep mode

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TIMjSLPEN**: TIMj clock enable during Sleep mode (j = 4 to 2)
0: TIMj clock disabled by the clock gating during Sleep mode
1: TIMj clock enabled by the clock gating during Sleep mode

10.5.33 RCC APB1 peripheral clock enable in Sleep mode register 2 (RCC_APB1SLPENR2)

Address offset: 0x0C8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SLPEN	Res.	Res.	Res.	LPTIM2SLPEN	Res.	Res.	Res.	Res.	Res.
						rw				rw					

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1SLPEN**: FDCAN1 clock enable during Sleep mode
0: FDCAN1 clocks disabled by the clock gating during Sleep mode
1: FDCAN1 clock enabled by the clock gating during Sleep mode

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SLPEN**: LPTIM2 clock enable during Sleep mode
0: LPTIM2 clock disabled by the clock gating during Sleep mode
1: LPTIM2 clock enabled by the clock gating during Sleep mode

Bits 4:0 Reserved, must be kept at reset value.

10.5.34 RCC APB2 peripheral clock enable in Sleep mode register (RCC_APB2SLPENR)

Address offset: 0x0CC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2SLPEN	Res.	Res.	USB1SLPEN	Res.	Res.	SAI1SLPEN	Res.	Res.	TIM17SLPEN	TIM16SLPEN	TIM15SLPEN
				rw			rw			rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1SLPEN	Res.	SPI1SLPEN	TIM1SLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **I3C2SLPEN**: I3C2 clock enable during Sleep mode
0: I3C2 clock disabled by the clock gating during Sleep mode
1: I3C2 clock enabled by the clock gating during Sleep mode

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **USB1SLPEN**: USB1 clock enable during Sleep mode
0: USB1 clock disabled by the clock gating during Sleep mode
1: USB1 clock enabled by the clock gating during Sleep mode

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SLPEN**: SAI1 clock enable during Sleep mode
0: SAI1 clock disabled by the clock gating during Sleep mode
1: SAI1 clock enabled by the clock gating during Sleep mode

Bits 20:19 Reserved, must be kept at reset value.

Bits 18:16 **TIMiSLPEN**: TIMi clock enable during Sleep mode (i = 17 to 15)
0: TIMi clock disabled during Sleep mode
1: TIMi clock enabled during Sleep mode

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1SLPEN**: USART1clock enable during Sleep mode
0: USART1clock disabled by the clock gating during Sleep mode
1: USART1clock enabled by the clock gating during Sleep mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SLPEN**: SPI1 clock enable during Sleep mode
0: SPI1 clock disabled by the clock gating during Sleep mode
1: SPI1 clock enabled by the clock gating during Sleep mode

Bit 11 **TIM1SLPEN**: TIM1 clock enable during Sleep mode
0: TIM1 clock disabled during Sleep mode
1: TIM1 clock enabled during Sleep mode

Bits 10:0 Reserved, must be kept at reset value.

10.5.35 RCC APB3 peripheral clock enable in Sleep mode register (RCC_APB3SLPENR)

Address offset: 0x0D0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP SLPEN	Res.	LPTIM4 SLPEN	LPTIM3 SLPEN	LPTIM1 SLPEN	Res.	Res.	Res.	I2C3SL PEN	LPUAR T1SLP EN	Res.	Res.	Res.	Res.	SYSCF GSLPE N	Res.
rw		rw	rw	rw				rw	rw					rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **COMPSLPEN**: COMP clock enable during Sleep mode
0: COMP clock disabled by the clock gating during Sleep mode
1: COMP clock enabled by the clock gating during Sleep mode

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LPTIMiSLPEN**: LPTIMi clock enable during Sleep mode (i = 4 to 3)
0: LPTIMi clock disabled during Sleep mode
1: LPTIMi clock enabled during Sleep mode

Bit 11 **LPTIM1SLPEN**: LPTIM1clock enable during Sleep mode
0: LPTIM1 clock disabled during Sleep mode
1: LPTIM1 clock enabled during Sleep mode

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3SLPEN**: I2C3 clock enable during Sleep mode
0: I2C3 clock disabled by the clock gating during Sleep mode
1: I2C3 clock enabled by the clock gating during Sleep mode

Bit 6 **LPUART1SLPEN**: LPUART1 clock enable during Sleep mode
0: LPUART1 clock disabled by the clock gating during Sleep mode
1: LPUART1 clock enabled by the clock gating during Sleep m

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGSLPEN**: SYSCFG clock enable during Sleep mode
0: SYSCFG clock disabled by the clock gating during Sleep mode
1: SYSCFG clock enabled by the clock gating during Sleep mode

Bit 0 Reserved, must be kept at reset value.

10.5.36 RCC AHB1 peripheral clock enable in Stop mode register (RCC_AHB1STPENR1)

Address offset: 0x0D8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1 STPEN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1 STPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMC FGSTP EN	Res.
rw							rw							rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASH STPEN	Res.	Res.	Res.	Res.	ADF1S TPEN	Res.	Res.	GPDM A1STP EN
							rw					rw			rw

Bit 31 **SRAM1STPEN**: SRAM1 clock enable during Stop mode
0: SRAM1 clock disabled by the clock gating during Stop mode
1: SRAM1 clock enabled by the clock gating during Stop mode

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1STPEN**: GTZC1 clock enable during Stop mode
0: GTZC1 clock disabled by the clock gating during Stop mode
1: GTZC1 clock enabled by the clock gating during Stop mode

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGSTPEN**: RAMCFG clock enable during Stop mode
0: RAMCFG clock disabled by the clock gating during Stop mode
1: RAMCFG clock enabled by the clock gating during Stop mode

Bits 16:9 Reserved, must be kept at reset value.

Bit 8 **FLASHSTPEN**: FLASH clock enable during Stop mode
0: FLASH clock disabled by the clock gating during Stop mode
1: FLASH clock enabled by the clock gating during Stop mode

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ADF1STPEN**: ADF1 clock enable during Stop mode.
0: ADF1 clock disabled by the clock gating during Stop mode
1: ADF1 clock enabled by the clock gating during Stop mode

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **GPDMA1STPEN**: GPDMA1 clock enable during Stop mode.
0: GPDMA1 clock disabled by the clock gating during Stop mode
1: GPDMA1 clock enabled by the clock gating during Stop mode

10.5.37 RCC AHB2 peripheral clock enable in Stop mode register 1 (RCC_AHB2STPENR1)

Address offset: 0x0DC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2 STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DAC1S TPEN	Res.	Res.	Res.	GPIOH STPEN	GPIOG STPEN	Res.	GPIOE STPEN	GPIOD STPEN	GPIOC STPEN	GPIOB STPEN	GPIOA STPEN
				rw				rw	rw		rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2STPEN**: SRAM2 clock enable during Stop mode
0: SRAM2 clock disabled by the clock gating during Stop mode
1: SRAM2 clock enabled by the clock gating during Stop mode

Bits 29:12 Reserved, must be kept at reset value.

Bit 11 **DAC1STPEN**: DAC1 clock enable during Stop mode
0: DAC1 clock disabled by the clock gating during Stop mode
1: DAC1 clock enabled by the clock gating during Stop mode

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:6 **GPIOiSTPEN**: I/O port i clock enable during Stop mode (i = H to G)
0: I/O port i clocks disabled by the clock gating during Stop mode
1: I/O port i clock enabled by the clock gating during Stop mode

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **GPIOiSTPEN**: I/O port i clock enable during Stop mode (i = E to A)
0: I/O port i clocks disabled by the clock gating during Stop mode
1: I/O port i clock enabled by the clock gating during Stop mode

10.5.38 RCC APB1 peripheral clock enable in Stop mode register 1 (RCC_APB1STPENR1)

Address offset: 0x0EC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RTCAPBSTPEN	VREFSTPEN	OPAMPSTPEN	Res.	Res.	Res.	Res.	I3C1STPEN	I2C2STPEN	I2C1STPEN	UART5STPEN	UART4STPEN	USART3STPEN	Res.	Res.
	rw	rw	rw					rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2STPEN	Res.	Res.	Res.	Res.	Res.	SPI3STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw						rw								

Bit 31 Reserved, must be kept at reset value.

Bit 30 **RTCAPBSTPEN**: RTC and TAMP APB clock enable during Stop mode
0: RTC and TAMP APB clock disabled by the clock gating during Stop mode
1: RTC and TAMP APB clock enabled by the clock gating during Stop mode

Bit 29 **VREFSTPEN**: VREFBUF clock enable during Stop mode
0: VREFBUF clock disabled by the clock gating during Stop mode
1: VREFBUF clock enabled by the clock gating during Stop mode

Bit 28 **OPAMPSTPEN**: OPAMP clock enable during Stop mode
0: OPAMP clock disabled by the clock gating during Stop mode
1: OPAMP clock enabled by the clock gating during Stop mode

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **I3C1STPEN**: I3C1 clock enable during Stop mode
0: I3C1 clock disabled by the clock gating during Stop mode
1: I3C1 clock enabled by the clock gating during Stop mode

Bit 22 **I2C2STPEN**: I2C2 clock enable during Stop mode
0: I2C2 clock disabled by the clock gating during Stop mode
1: I2C2 clock enabled by the clock gating during Stop mode

Bit 21 **I2C1STPEN**: I2C1 clock enable during Stop mode
0: I2C1 clock disabled by the clock gating during Stop mode
1: I2C1 clock enabled by the clock gating during Stop mode

Bit 20 **UART5STPEN**: UART5 clock enable during Stop mode
0: UART5 clock disabled by the clock gating during Stop mode
1: UART5 clock enabled by the clock gating during Stop mode

Bit 19 **UART4STPEN**: UART4 clock enable during Stop mode
0: UART4 clock disabled by the clock gating during Stop mode
1: UART4 clock enabled by the clock gating during Stop mode

Bit 18 **USART3STPEN**: USART3 clock enable during Stop mode
0: USART3 clock disabled by the clock gating during Stop mode
1: USART3 clock enabled by the clock gating during Stop mode

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **SPI2STPEN**: SPI2 clock enable during Stop mode
0: SPI2 clock disabled by the clock gating during Stop mode
1: SPI2 clock enabled by the clock gating during Stop mode

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **SPI3STPEN**: SPI3 clock enable during Stop mode
0: SPI3 clock disabled by the clock gating during Stop mode
1: SPI3 clock enabled by the clock gating during Stop mode

Bits 7:0 Reserved, must be kept at reset value.

10.5.39 RCC APB1 peripheral clock enable in Stop mode register 2 (RCC_APB1STPENR2)

Address offset: 0x0F0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2 STPEN	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2STPEN**: LPTIM2 clock enable during Stop mode
0: LPTIM2 clock disabled by the clock gating during Stop mode
1: LPTIM2 clock enabled by the clock gating during Stop mode

Bits 4:0 Reserved, must be kept at reset value.

10.5.40 RCC APB2 peripheral clock enable in Stop mode register (RCC_APB2STPENR)

Address offset: 0x0F4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2STPEN	Res.	Res.	USB1STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw			rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1STPEN	Res.	SPI1STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **I3C2STPEN**: I3C2 clock enable during Stop mode
0: I3C2 clock disabled by the clock gating during Stop mode
1: I3C2 clock enabled by the clock gating during Stop mode

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **USB1STPEN**: USB1 clock enable during Stop mode
0: USB1 clock disabled by the clock gating during Stop mode
1: USB1 clock enabled by the clock gating during Stop mode

Bits 23:15 Reserved, must be kept at reset value.

Bit 14 **USART1STPEN**: USART1clock enable during Stop mode
0: USART1 clock disabled by the clock gating during Stop mode
1: USART1 clock enabled by the clock gating during Stop mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1STPEN**: SPI1 clock enable during Stop mode
0: SPI1 clock disabled by the clock gating during Stop mode
1: SPI1 clock enabled by the clock gating during Stop mode

Bits 11:0 Reserved, must be kept at reset value.

10.5.41 RCC APB3 peripheral clock enable in Stop mode register (RCC_APB3STPENR)

Address offset: 0x0F8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word, and byte access

This register only configures the clock gating, not the clock source itself. When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPSTPEN	Res.	LPTIM4STPEN	LPTIM3STPEN	LPTIM1STPEN	Res.	Res.	Res.	I2C3STPEN	LPUART1STPEN	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw				rw	rw						

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **COMPSTPEN**: COMP clock enable during Stop mode
0: COMP clock disabled by the clock gating during Stop mode
1: COMP clock enabled by the clock gating during Stop mode

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LPTIMiSTPEN**: LPTIMi clock enable during Stop mode (i = 4 to 3)
0: LPTIMi clock disabled during Stop mode
1: LPTIMi clock enabled during Stop mode

Bit 11 **LPTIM1STPEN**: LPTIM1clock enable during Stop mode
0: LPTIM1 clock disabled during Stop mode
1: LPTIM1 clock enabled during Stop mode

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3STPEN**: I2C3 clock enable during Stop mode
0: I2C3 clock disabled by the clock gating during Stop mode
1: I2C3 clock enabled by the clock gating during Stop mode

Bit 6 **LPUART1STPEN**: LPUART1 clock enable during Stop mode
0: LPUART1 clock disabled by the clock gating during Stop mode
1: LPUART1 clock enabled by the clock gating during Stop m

Bits 5:0 Reserved, must be kept at reset value.

10.5.42 RCC peripheral independent clock configuration register 1 (RCC_CCIPR1)

Address offset: 0x100

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMICSEL[2:0]			USB1SEL	ICLKSEL[1:0]		Res.	FDCAN1SEL	SYSTICKSEL[1:0]		Res.	SPI1SEL	LPTIM2SEL[1:0]		Res.	SPI2SEL
rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	I3C2SEL	Res.	I2C2SEL	Res.	I2C1SEL	Res.	I3C1SEL	Res.	UART5SEL	Res.	UART4SEL	Res.	USART3SEL	Res.	USART1SEL
	rw		rw		rw		rw		rw		rw		rw		rw

Bits 31:29 **TIMICSEL[2:0]**: Clock sources for TIM16,TIM17, and LPTIM2 internal input capture

When the TIMICSEL[2] bit is set, TIM16, TIM17, and LPTIM2 internal input capture can be connected either to HSI/256, MSI/4, or MSI/1024. Depending on TIMICSEL[1:0] value, MSI is either MSIK or MSIS.

When TIMICSEL[2] is cleared, the HSI, MSIK, and MSIS clock sources cannot be selected as TIM16, TIM17, or LPTIM2 internal input capture.

000-011: HSI, MSIK and MSIS dividers disabled

100: HSI/256, MSIS/1024 and MSIS/4 are generated and can be selected by TIM16, TIM17, and LPTIM2 as internal input capture.

101: HSI/256, MSIS/1024 and MSIK/4 are generated and can be selected by TIM16, TIM17, and LPTIM2 as internal input capture.

110: HSI/256, MSIK/1024 and MSIS/4 are generated and can be selected by TIM16, TIM17, and LPTIM2 as internal input capture.

111: HSI/256, MSIK/1024 and MSIK/4 are generated and can be selected by TIM16, TIM17, and LPTIM2 as internal input capture.

Note: The clock division must be disabled (TIMICSEL[2:0] = 0xx) before selecting or changing a clock source division.

Bit 28 **USB1SEL**: USB1 kernel clock prescaler selection

This bit is used to configure the division factor of the USB1 kernel clock source selected with ICLKSEL.

0: usbsdmmc_iclk selected

1: usbsdmmc_iclk/2 selected

Bits 27:26 **ICLKSEL[1:0]**: Intermediate clock source selection

These bits are used to select the clock source for USB and SDMMC (usbsdmmc_iclk).

00: HSI48 selected

01: MSIK selected

10: HSE selected

11: SYSCLK selected

Bit 25 Reserved, must be kept at reset value.

Bit 24 **FDCAN1SEL**: FDCAN1 kernel clock source selection

0: SYSCLK selected

1: MSIK selected

Bits 23:22 **SYSTICKSEL[1:0]**: SysTick clock source selection
00: HCLK/8 selected
01: LSI selected
10: LSE selected
11: reserved
Note: When LSE or LSI is selected, the AHB frequency must be at least four times higher than the LSI or LSE frequency. In addition, a jitter up to one HCLK cycle is introduced, due to the LSE or LSI sampling with HCLK in the SysTick circuitry.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **SPI1SEL**: SPI1 kernel clock source selection
0: PCLK2 selected
1: MSIK selected
Note: The SPI1 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bits 19:18 **LPTIM2SEL[1:0]**: Low-power timer 2 kernel clock source selection
00: PCLK1 selected
01: LSI selected
10: HSI16 selected
11: LSE selected
Note: The LPTIM2 is functional in Stop 0 or Stop 1 mode only when the kernel clock is LSI, LSE, or HSI16 if HSIKERON = 1.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **SPI2SEL**: SPI2 kernel clock source selection
0: PCLK1 selected
1: MSIK selected
Note: The SPI2 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **I3C2SEL**: I3C2 kernel clock source selection
0: PCLK2 selected
1: MSIK selected
Note: The I3C2 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **I2C2SEL**: I2C2 kernel clock source selection
0: PCLK1 selected
1: MSIK selected
Note: The I2C2 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **I2C1SEL**: I2C1 kernel clock source selection
0: PCLK1 selected
1: MSIK selected
Note: The I2C1 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **I3C1SEL**: I3C1 kernel clock source selection
0: PCLK1 selected
1: MSIK selected
Note: The I3C1 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **UART5SEL**: UART5 kernel clock source selection
0: PCLK1 selected
1: HSI16 selected
Note: The UART5 is functional in Stop 0 or Stop 1 mode only when the kernel clock is HSI16.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **UART4SEL**: UART4 kernel clock source selection
0: PCLK1 selected
1: HSI16 selected
Note: UART4 is functional in Stop 0 or Stop 1 mode only when the kernel clock is HSI16.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **USART3SEL**: USART3 kernel clock source selection
0: PCLK1 selected
1: HSI16 selected
Note: USART3 is functional in Stop 0 or Stop 1 mode only when the kernel clock is HSI16.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **USART1SEL**: USART1 kernel clock source selection
0: PCLK2 selected
1: HSI16 selected
Note: USART1 is functional in Stop 0 or Stop 1 mode only when the kernel clock is HSI16.

10.5.43 RCC peripheral independent clock configuration register 2 (RCC_CCIPR2)

Address offset: 0x104

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPISEL	DAC1SHSEL	Res.	ADCDACSEL[1:0]	
											rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCDACPRE[3:0]				RNGSEL	Res.	Res.	Res.	Res.	SAI1SEL[1:0]		Res.	SPI3SEL	Res.	ADF1SEL[1:0]	
rw	rw	rw	rw	rw					rw	rw		rw		rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPISEL**: OCTOSPI1 kernel clock source selection
0: SYSCLK selected
1: MSIK selected

Bit 19 **DAC1SHSEL**: DAC1 sample and hold clock source selection
0: LSE selected
1: LSI selected

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **ADCDACSEL[1:0]**: ADC12 and DAC1 intermediate kernel clock source selection
These bits are used to select the clock source for the ADC12 and DAC1 (adcdac_iclk)).
00: HCLK selected
01: HSE selected
10: MSIK selected
11: Reserved

Bits 15:12 **ADCDACPRE[3:0]**: ADC12 and DAC1 kernel clock prescaler
These bits are used to configure the division factor of the ADC12 and DAC1 kernel clock source selected with ADCDACSEL.
0000: adcdac_iclk
0001-0111: adcdac_iclk/2
1000: adcdac_iclk/4
1001: adcdac_iclk/8
1010: adcdac_iclk/16
1011: adcdac_iclk/32
1100: adcdac_iclk/64
1101: adcdac_iclk/128
1110: adcdac_iclk/256
1111: adcdac_iclk/512

Bit 11 **RNGSEL**: RNG kernel clock source selection
0: HSI48 selected
1: MSIK selected

Bits 10:7 Reserved, must be kept at reset value.

Bits 6:5 **SAI1SEL[1:0]**: SAI1 kernel clock source selection
00: MSIK selected
01: input pin AUDIOCLK selected
10: HSE clock selected
11: Reserved
Note: If the selected clock is the external clock and this clock is stopped, a switch to another clock is impossible.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SPI3SEL**: SPI3 kernel clock source selection
0: PCLK1 selected
1: MSIK selected
Note: The SPI3 is functional in Stop 0 or Stop 1 mode only when the kernel clock is MSIK.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **ADF1SEL[1:0]**: ADF1 kernel clock source selection
00: HCLK
01: Input pin AUDIOCLK selected
10: MSIK clock selected
11: SAI1 kernel clock selected
Note: If the selected clock is the external clock and this clock is stopped, a switch to another clock is impossible.

10.5.44 RCC peripheral independent clock configuration register 3 (RCC_CCIPR3)

Address offset: 0x108

Reset value: 0x0000 0000

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPTIM1SEL[1:0]		LPTIM34SEL[1:0]		Res.	I2C3SEL	Res.	Res.	Res.	Res.	LPUART1SEL[1:0]	
				rw	rw	rw	rw		rw					rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 **LPTIM1SEL[1:0]**: LPTIM1 kernel clock source selection
These bits are used to select the LPTIM1 kernel clock source.
00: MSIK clock selected
01: LSI selected
10: HSI16 selected
11: LSE selected
Note: The LPTIM1 is functional in Stop 0, Stop 1, and Stop 2 modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1, or MSIK with MSIKERON = 1.

Bits 9:8 **LPTIM34SEL[1:0]:** LPTIM3 and LPTIM4 kernel clock source selection
00: MSIK clock selected
01: LSI selected
10: HSI16 selected
11: LSE selected
Note: LPTIM3 and LPTIM4 are functional in Stop 0, Stop 1, and Stop 2 modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1, or MSIK with MSIKERON = 1.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **I2C3SEL**: I2C3 kernel clock source selection
0: PCLK3 selected
1: MSIK selected
Note: The I2C3 is functional in Stop 0, Stop 1, and Stop 2 modes only when the kernel clock is MSIK.

Bits 5:2 Reserved, must be kept at reset value.

Bits 1:0 **LPUART1SEL[1:0]**: LPUART1 kernel clock source selection
00: PCLK3 selected
01: HSI16 selected
10: LSE selected
11: MSIK selected
Note: The LPUART1 is functional in Stop 0, Stop 1, and Stop 2 modes only when the kernel clock is HSI16, LSE, or MSIK.

10.5.45 RCC backup domain control register (RCC_BDCR)

Address offset: 0x110

Reset value: 0xUUUU UUUU

Backup domain reset value: 0x0000 X000

Reset by backup domain reset, except LSCOSEL, LSCOEN, and BDRST that are reset only by backup domain power-on reset. LSESYSEN and LSESYSRDY are reset by power-on reset.

This register bits are outside of the core domain. As a result, after reset, these bits are write-protected, and DBP in PWR_BDCR must be set before these can be modified (see *Section 10.1: RCC introduction*). These bits (except LSESYSEN, LSESYSRDY, LSCOSEL, LSCOEN, and BDRST) are only reset after a backup domain reset (see *Backup domain reset*). Any internal or external reset does not have any effect on these bits.

Access: 0 ≤ wait state ≤ 3; word, half-word, and byte access

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
						rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	LSEGFON	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL**: Low-speed clock output selection
0: LSI selected
1: LSE selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
0: LSCO disabled
1: LSCO enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: Backup domain software reset
0: No effect
1: Reset the entire backup domain

Bit 15 **RTCEN**: RTC and TAMP clock enable
0: RTC and TAMP clock disabled
1: RTC and TAMP clock enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LSEGFON**: LSE clock glitch filter enable
This bit can be written only when the LSE is disabled (LSEON = 0 and LSERDY = 0).
0: LSE glitch filter disabled
1: LSE glitch filter enabled

Bit 11 **LSESYSRDY**: LSE system clock (LSESYS) ready

Set and cleared by hardware to indicate when the LSE system clock is stable.When the LSESYSEN bit is set, the LSESYSRDY flag is set after two LSE clock cycles.

The LSE clock must be already enabled and stable (LSEON and LSERDY are set).

When the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.

0: LSESYS clock not ready

1: LSESYS clock ready

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: RTC and TAMP clock source selection

Once the RTC and TAMP clock source has been selected, it cannot be changed anymore unless the backup domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The BDRST bit can be used to reset them.

00: No clock selected

01: LSE selected

10: LSI selected

11: HSE/32 selected

Bit 7 **LSESYSEN**: LSE system clock (LSESYS) enable

Set by software to enable always the LSE system clock generated by RCC, which can be used by any peripheral when its source clock is the LSE, or at system level if one of LSCO, MCO, or MSI PLL mode is needed.

0: LSE can be used only for RTC, TAMP, and CSS on LSE.

1: LSE can be used by any other peripheral or function.

Bit 6 **LSECSSD**: CSS on LSE failure detection

Set by hardware to indicate when a failure is detected by the CCS on the external 32 kHz oscillator (LSE). This flag is cleared when LSECSSON = 0.

0: No failure detected on LSE

1: Failure detected on LSE

Bit 5 **LSECSSON**: CSS on LSE enable

This bit must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.

Clearing LSECSSON also clears the LSECSSD bit.

0: CSS on LSE OFF

1: CSS on LSE ON

Bits 4:3 **LSEDRV[1:0]:** LSE oscillator drive capability

This bitfield can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

00: 'Xtal mode' lower driving capability

01: 'Xtal mode' medium-low driving capability

10: 'Xtal mode' medium-high driving capability

11: 'Xtal mode' higher driving capability

Note: The oscillator is in 'Xtal mode' when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

This bit can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.

0: LSE oscillator not ready

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

0: LSE oscillator OFF

1: LSE oscillator ON

10.5.46 RCC control/status register (RCC_CSR)

Address offset: 0x114

Reset value: 0x0C00 5500

Reset by system reset, except reset flags by power reset only.

Access: 0 ≤ wait state ≤ 3; word, half-word, and byte access

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MSISDIVS[1:0]		Res.	Res.	MSIKDIVS[1:0]		Res.	Res.	Res.	Res.	Res.	LSIPREDIV	LSIRDY	LSION
		rw	rw			rw	rw						rw	rw	rw

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to Stop, Standby, or Shutdown mode entry, whereas the corresponding NRST_STOP, NRST_STBY, or NRST_SHDW option bit is cleared. Cleared by writing to the RMVF bit.

0: No illegal low-power mode reset occurred.

1: Illegal low-power mode reset occurred.

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs. Cleared by writing to the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent watchdog reset flag

Set by hardware when an independent watchdog reset domain occurs.

Cleared by writing to the RMVF bit.

0: No independent watchdog reset occurred.

1: Independent watchdog reset occurred.

Bit 28 **SFTRSTF**: Software reset flag

Set by hardware when a software reset occurs. Cleared by writing to the RMVF bit.

0: No software reset occurred.

1: Software reset occurred.

Bit 27 **BORRSTF**: BOR flag
Set by hardware when a BOR occurs. Cleared by writing to the RMVF bit.
0: No BOR occurred.
1: BOR occurred.

Bit 26 **PINRSTF**: NRST pin reset flag
Set by hardware when a reset from the NRST pin occurs. Cleared by writing to the RMVF bit.
0: No reset from NRST pin occurred.
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF**: Option-byte loader reset flag
Set by hardware when a reset from the option-byte loading occurs.
Cleared by writing to the RMVF bit.
0: No reset from option-byte loading occurred.
1: Reset from option-byte loading occurred.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flag
0: No effect
1: Clear the reset flags.

Bits 22:14 Reserved, must be kept at reset value.

Bits 13:12 **MSISDIVS[1:0]**: MSIS oscillator division after Standby mode
Set by software to choose the MSIS frequency at Standby mode exit, and until MSIRGSEL is set. After a system reset, a power-on reset, or when exiting Shutdown mode, range 5 (12MHz) is always selected. MSISDIVS can be written only when MSIRGSEL = 1.
01: Range 5 around 12 MHz (reset value)
10: Range 6 around 6 MHz
11: Range 7 around 3 MHz
Others: reserved
Note: Changing the MSISDIVS does not change the current MSIS frequency.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **MSIKDIVS[1:0]**: MSIK oscillator division after Standby mode
Set by software to choose the MSIK frequency at Standby mode exit and until MSIRGSEL is set. After a system reset, a power-on reset or when exiting Shutdown mode, range 5 (12MHz) is always selected. MSIKDIVS can be written only when MSIRGSEL = 1.
01: Range 5 around 12 MHz (reset value)
10: Range 6 around 6 MHz
11: Range 7 around 3 MHz
Others: reserved
Note: Changing the MSIKDIVS does not change the current MSIK frequency.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **LSIPREDIV**: Low-speed clock divider configuration
This bit can be written only when the LSI is disabled (LSION = 0 and LSIRDY = 0). If the LSI was previously enabled, it is necessary to wait for LSIRDY = 0 after clearing LSION bit (synchronization time for LSI to be really disabled), before writing LSIPREDIV. LSIPREDIV cannot be changed if the LSI is used by the IWDG, by the RTC, or by the TAMP.
0: LSI not divided
1: LSI divided by 128

Bit 1 **LSIRDY**: LSI oscillator ready

Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles. This bit is set when the LSI is used by IWDG, RTC, or TAMP, even if LSION = 0.

0: LSI oscillator not ready

1: LSI oscillator ready

Bit 0 **LSION**: LSI oscillator enable

0: LSI oscillator OFF

1: LSI oscillator ON

10.5.47 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x130

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be written only by a secure privileged access if SPRIV = 1, and by a secure privileged or unprivileged access if SPRIV = 0. A nonsecure write access generates an illegal access event and data is not written. This register can be read by secure or nonsecure, privileged or unprivileged access.When the system is not secure (TZEN = 0), this register is read as 0, and the register write is ignored.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVFSEC	HSI48SEC	ICLKSEC	Res.	Res.	BOOSTSEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	MSISEC	HSESEC	HSISEC
			rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RMVFSEC**: Remove reset flag security

0: Nonsecure

1: Secure

Bit 11 **HSI48SEC**: HSI48 clock configuration and status bit security

0: Nonsecure

1: Secure

Bit 10 **ICLKSEC**: Intermediate clock source selection security

0: Nonsecure

1: Secure

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **BOOSTSEC**: EPOD booster configuration bit security

0: Nonsecure

1: Secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bit security
0: Nonsecure
1: Secure

Bit 5 **SYSCLKSEC**: SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration security
0: Nonsecure
1: Secure

Bit 4 **LSESEC**: LSE clock configuration and status bit security
0: Nonsecure
1: Secure

Bit 3 **LSISEC**: LSI clock configuration and status bit security
0: Nonsecure
1: Secure

Bit 2 **MSISEC**: MSI clock configuration and status bit security
0: Nonsecure
1: Secure

Bit 1 **HSESEC**: HSE clock configuration, status bits, and HSE_CSS security
0: Nonsecure
1: Secure

Bit 0 **HSISEC**: HSI clock configuration and status bit security
0: Nonsecure
1: Secure

10.5.48 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x134

Reset value: 0x0000 0000

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

Access: no wait state; word, half-word, and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: RCC nonsecure function privilege configuration
This bit can be written only by privileged access, secure, or nonsecure.
0: Read and write to RCC nonsecure functions can be done by privileged or unprivileged access.
1: Read and write to RCC nonsecure functions can be done by privileged access only.

Bit 0 **SPRIV**: RCC secure function privilege configuration
This bit can be written only by a secure privileged access.
0: Read and write to RCC secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC secure functions can be done by privileged access only.

10.5.49 RCC register map

Table 101. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON	HSI48RDY	HSI48ON	HSIRDY	HSIKERON	HSION	MSIPL0RDY	MSIPL1RDY	MSIPL0FAST	MSIPL1FAST	MSIPL0EN	MSIPL1EN	MSIKKRDY	MSIKON	MSISRDY	MSIKERON	MSISON
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
0x004	Reserved	Reserved																															
0x008	RCC_ICSCR1	MSISSEL	MSISDIV[1:0]		MSIKESEL	MSIKDIV[1:0]		MSIPPLN[1:0]		MSIRGSEL	MSIBIAS	MSIPL01SEL	MSIPLL1SEL	MSIHSINDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSICAL0[5:0]						MSICAL1[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0								X	X	X	X	X	X	X	X	X	X	X	X
0x00C	RCC_ICSCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSITRIM0[5:0]						MSITRIM1[5:0]					
	Reset value																					1	0	0	0	0	0	1	0	0	0	0	0
0x010	RCC_ICSCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[4:0]					Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value												1	0	0	0	0					X	X	X	X	X	X	X	X	X	X	X	X
0x014	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x018	Reserved	Reserved																															
0x01C	RCC_CFGR1	Res.	MCOPRE [2:0]			MCOSEL [3:0]				Res.	MCO2PRE [2:0]			MCO2SEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPKERWUCK	STOPWUCK	SWS[1:0]		SW[1:0]	
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0											0	0	0	0	0	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE2 [2:0]			Res.	PPRE1 [2:0]			HPRE[3:0]			
	Reset value																						0	0	0		0	0	0	0	0	0	0
0x024	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res	Res	Res.	Res.	Res	Res.	Res.	Res.	Res.	PPRE3 [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x028	RCC_CFRGGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTDIV [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTSEL [1:0]	
	Reset value																	0	0	0	0											0	0
0x02C-0x04C	Reserved	Reserved																															

Table 101. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSIE	MSIKRDYIE		MSIPLLHSUIE	MSIPLLUIE	MSIPLL0RDYIE	MSIPLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	MSISRDYIE	LSERDYIE	LSIRDYIE
	Reset value																				0	0		0	0	0	0	0	0	0	0	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSF	MSIKRDYF	CSSF	MSIPLLHSUF	MSIPLLUF	MSIPLL0RDYF	MSIPLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	MSISRDYF	LSERDYF	LSIRDYF
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSC	MSIKRDYC	CSSC	MSIPLLHSUC	MSIPLLUC	MSIPLL0RDYC	MSIPLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	MSISRDYC	LSERDYC	LSIRDYC
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	Reserved	Reserved																															
0x060	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGRST	TSCRST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RST	Res.	Res.	GPDMA1RST
	Reset value															0	0				0									0			0
0x064	RCC_ AHB2RSTR1	Res.	Res.	Res.	Res.	SDMMC1RS	Res.	Res.	Res.	Res.	Res.	CCBRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST	Res.	Res.	Res.	Res.	DAC1RST	ADC12RST	Res.	Res.	GPIOHRST	GPIOGRST	Res.	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value					0						0	0	0	0	0	0					0	0			0	0		0	0	0	0	0
0x068	RCC_ AHB2RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1RST	Res.	Res.	Res.	Res.
	Reset value																												0				
0x06C-0x070	Reserved	Reserved																															
0x074	RCC_ APB1RSTR1	Res.	Res.	VREFRST	OPAMPRST	Res.	Res.	Res.	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	Res.	Res.	Res.	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	SPI3RST	Res.	Res.	TIM7RST	TIM6RST	Res.	TIM4RST	TIM3RST	TIM2RST
	Reset value			0	0				0	0	0	0	0	0	0				0						0			0	0		0	0	0
0x078	RCC_ APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1RST	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	Res.
	Reset value																							0				0					
0x07C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	I3C2RST	Res.	Res.	USB1RST	Res.	Res.	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	Res.	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0			0			0			0	0	0		0		0	0												

Table 101. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	RCC_APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPRST	Res.	LPTIM4RST	LPTIM3RST	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RS	Res.	Res.	Res.	Res.	SYSCFGRST	Res.
	Reset value																	0		0	0	0				0	0					0	
0x084	Reserved	Reserved																															
0x088	RCC_AHB1ENR1	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	TSCEN	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	ADF1EN	Res.	Res.	GPDMA1EN
	Reset value	1							0							0	0				0				1					0			0
0x08C	RCC_AHB2ENR1	Res.	SRAM2EN	Res.	Res.	SDMMC1EN	Res.	Res.	Res.	Res.	Res.	CCBEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN	Res.	Res.	Res.	Res.	DAC1EN	ADC12EN	Res.	Res.	GPIOHEN	GPIOGEN	Res.	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value		1			0						0	0	0	0	0	0					0	0			0	0		0	0	0	0	0
0x090	RCC_AHB2ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1EN	Res.	Res.	Res.	Res.
	Reset value																												0				
0x094	RCC_AHB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWREN	Res.	Res.
	Reset value																														0		
0x098	Reserved	Reserved																															
0x09C	RCC_APB1ENR1	Res.	RTCAPBEN	VREFEN	OPAMPEN	Res.	Res.	Res.	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	Res.	Res.	Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	SPI3EN	Res.	Res.	TIM7EN	TIM6EN	Res.	TIM4EN	TIM3EN	TIM2EN
	Reset value		0	0	0				0	0	0	0	0	0	0				0			0			0			0	0		0	0	0
0x0A0	RCC_APB1ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1EN	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.
	Reset value																							0				0					
0x0A4	RCC_APB2ENR	Res.	Res.	Res.	Res.	I3C2EN	Res.	Res.	USBEN	Res.	Res.	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0			0			0			0	0	0		0		0	0											
0x0A8	RCC_APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	LPTIM4EN	LPTIM3EN	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	Res.	Res.	Res.	Res.	SYSCFGEN	Res.
	Reset value																	0		0	0	0				0	0					0	
0x0AC	Reserved	Reserved																															
0x0B0	RCC_ AHB1SLPENR1	SRAM1SLPEN	Res.	ICACHESLPEN	Res.	Res.	Res.	Res.	GTZC1SLPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSLPEN	TSCSLPEN	Res.	Res.	Res.	CRCSLPEN	Res.	Res.	Res.	FLASHSLPEN	Res.	Res.	Res.	Res.	ADF1SLPEN	Res.	Res.	GPDMA1SLPEN
	Reset value	1		1					1							1	1				1				1					1			1

Table 101. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B4	RCC_ AHB2SLPENR1	Res.	SRAM2SLPEN	Res.	Res.	SDMMC1SLPEN	Res.	Res.	Res.	Res.	Res.	CCBSLPEN	SAESSLPEN	PKASLPEN	RNGSLPEN	HASHSLPEN	AESSLPEN	Res.	Res.	Res.	Res.	DAC1SLPEN	ADC12SLPEN	Res.	Res.	GPIOHSLPEN	GPIOGSLPEN	Res.	GPIOESLPEN	GPIODSLPEN	GPIOCSLPEN	GPIOBSLPEN	GPIOASLPEN
	Reset value		1			1						1	1	1	1	1	1					1	1			1	1		1	1	1	1	1
0x0B8	RCC_ AHB2SLPENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1SLPEN	Res.	Res.	Res.	Res.
	Reset value																												1				
0x0BC	RCC_ AHB1SLPENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWRSLPEN	Res.	Res.
	Reset value																														1		
0x0C0	Reserved	Reserved																															
0x0C4	RCC_ APB1SLPENR1	Res.	RTCAPBLPEN	VREFLPEN	OPAMPLPEN	Res.	Res.	Res.	CRSSLPEN	I3C1SLPEN	I2C2SLPEN	I2C1SLPEN	UART5SLPEN	UART4SLPEN	USART3SLPEN	Res.	Res.	Res.	SPI2SLPEN	Res.	Res.	WWDGSLPEN	Res.	Res.	SPI3SLPEN	Res.	Res.	TIM7SLPEN	TIM6SLPEN	Res.	TIM4SLPEN	TIM3SLPEN	TIM2SLPEN
	Reset value		1	1	1				1	1	1	1	1	1	1				1			1			1			1	1		1	1	1
0x0C8	RCC_ APB1SLPENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SLPEN	Res.	Res.	Res.	LPTIM2SLPEN	Res.	Res.	Res.	Res.	Res.
	Reset value																							1				1					
0x0CC	RCC_ APB2SLPENR	Res.	Res.	Res.	Res.	I3C2SLPEN	Res.	Res.	USB1SLPEN	Res.	Res.	SAI1SLPEN	Res.	Res.	TIM17SLPEN	TIM16SLPEN	TIM15SLPEN	Res.	USART1SLPEN	Res.	SPI1SLPEN	TIM1SLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					1			1			1			1	1	1		1		1	1											
0x0D0	RCC_ APB3SLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPSLPEN	Res.	LPTIM4SLPEN	LPTIM3SLPEN	LPTIM1SLPEN	Res.	Res.	Res.	I2C3SLPEN	LPUART1SLPEN	Res.	Res.	Res.	Res.	SYSCFGSLPEN	Res.
	Reset value																	1		1	1	1				1	1					1	
0x0D4	Reserved	Reserved																															
0x0D8	RCC_ AHB1STPENR1	SRAM1STPEN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1STPEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMPCFGSTPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHSTPEN	Res.	Res.	Res.	Res.	ADF1STPEN	Res.	Res.	GPDMA1STPEN
	Reset value	1							1							1									1					1			1

Table 101. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0DC	RCC_ AHB2STPENR1	Res.	SRAM2STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1STPEN	Res.	Res.	Res.	GPIOHSTPEN	GPIOGSTPEN	Res.	GPIOESTPEN	GPIODSTPEN	GPIOCSTPEN	GPIOBSTPEN	GPIOASTPEN
	Reset value		1																			1				1	1		1	1	1	1	1
0x0E0-0xE8	Reserved	Reserved																															
0x0EC	RCC_ APB1STPENR1	Res.	RTCAPBSTPEN	VREFSTPEN	OPAMPSTPEN	Res.	Res.	Res.	Res.	I3C1STPEN	I2C2STPEN	I2C1STPEN	UART5STPEN	UART4STPEN	USART3STPEN	Res.	Res.	Res.	SPI2STPEN	Res.	Res.	Res.	Res.	Res.	SPI3STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		1	1	1					1	1	1	1	1	1				1						1								
0x0F0	RCC_ APB1STPENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2STPEN	Res.	Res.	Res.	Res.	Res.
	Reset value																											1					
0x0F4	RCC_ APB2STPENR	Res.	Res.	Res.	Res.	I3C2STPEN	Res.	Res.	LPTIM2STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USAR1STPEN	Res.	SPI1STPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					1			1										1		1												
0x0F8	RCC_ APB3STPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPSTPEN	Res.	LPTIM4STPEN	LPTIM3STPEN	LPTIM1STPEN	Res.	Res.	Res.	I2C3STPEN	LPUART1STPEN	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	1		1	1	1				1	1						
0x0FC	Reserved	Reserved																															
0x100	RCC_CCIPR1	TIMICSEL [2:0]			USB1SEL	ICLKSEL [1:0]		Res.	FDCANSEL	SYSTICKSEL [1:0]		Res.	SPI1SEL	LPTIM2SEL [1:0]		Res.	SPI2SEL	Res.	I3C2SEL	Res.	I2C2SEL	Res.	I2C1SEL	Res.	I3C1SEL	Res.	UART5SEL	Res.	UART4SEL	Res.	UART3SEL	Res.	USART1SEL
	Reset value	0	0	0	0	0	0		0	0	0		0	0	0		0		0		0		0		0		0		0		0		0
0x104	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPISEL	DAC1SHSEL	Res.	ADCDACSEL [1:0]		ADCDACPRE [3:0]				RNGSEL	Res.	Res.	Res.	Res.	SAI1SEL[1:0]		Res.	SPI3SEL	Res.	ADFI1SEL [1:0]	
	Reset value												0	0		0	0	0	0	0	0	0					0	0		0		0	0
0x108	RCC_CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SEL [1:0]		LPTIM34SEL [1:0]		Res.	I2C3SEL	Res.	Res.	Res.	Res.	LPUART1SEL [1:0]	
	Reset value																					0	0	0	0		0					0	0
0x110	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	LSEGFON	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
	Reset value							0	0								0	0			0	X		X		0	0	0	0		0	0	0

Table 101. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x114	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSISDIVS [1:0]		Res.	Res.	MSIKDIVS [1:0]		Res.	Res.	Res.	Res.	Res.	LSIPREDIV	LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0		0										0	1			0	1						0	0	0
0x118-0x12C	Reserved	Reserved																															
0x130	RCC_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	HSI48SEC	ICLKSEC	Res.	Res.	BOOSTSEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	MSISEC	HSESEC	HSISEC
	Reset value																				0	0	0			0	0	0	0	0	0	0	0
0x134	RCC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0

Refer to *Section 2.3* for the register boundary addresses.

11 Clock recovery system (CRS)

11.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

11.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 11.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

11.3 CRS implementation

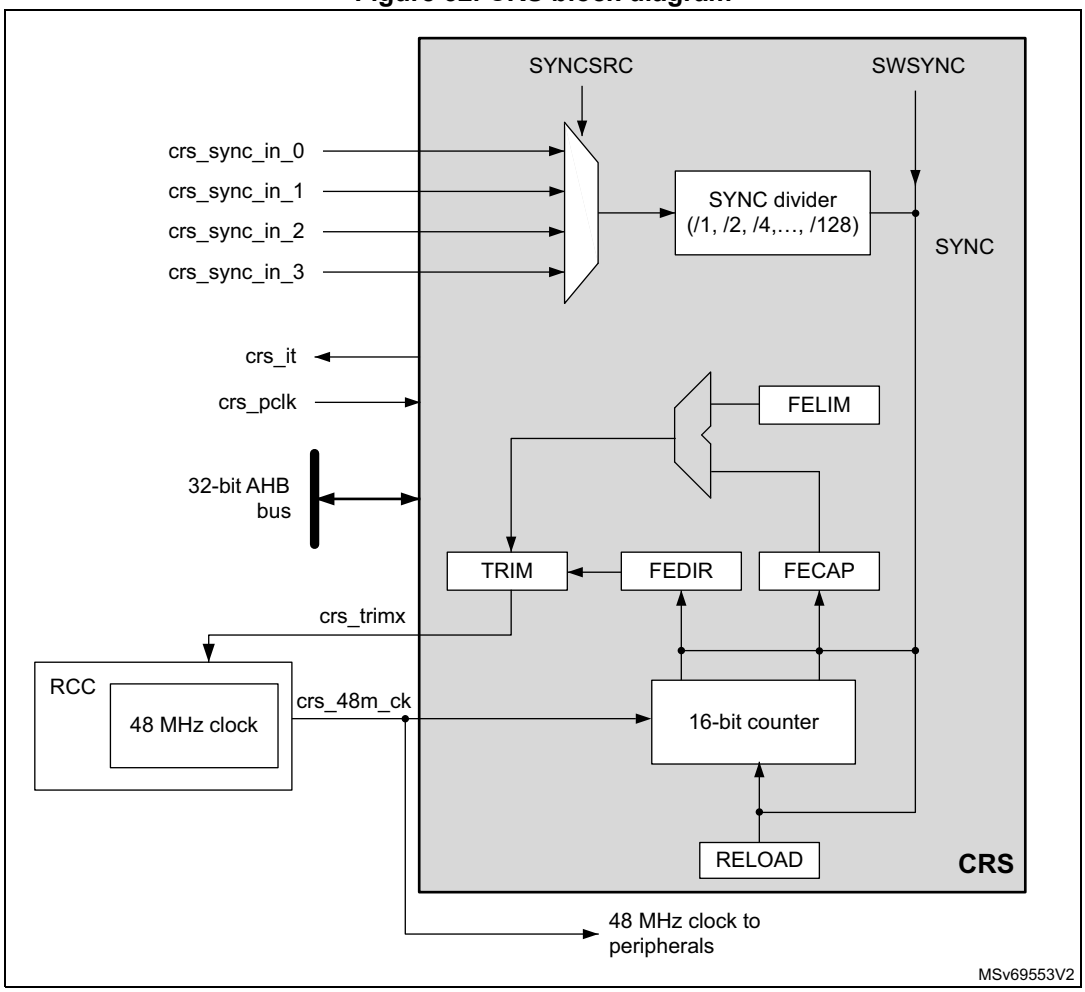
Table 102. CRS features

Feature	CRS
TRIM width	7 bits

11.4 CRS functional description

11.4.1 CRS block diagram

Figure 32. CRS block diagram



11.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 103. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:6]	Digital output	HSI48 oscillator smooth trimming value

Table 103. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 104. CRS interconnection

Internal signal name	Source
crs_sync_in_0	CRS_SYNC
crs_sync_in_1	LSE clock
crs_sync_in_2	USB SOF
crs_sync_in_3	Reserved

11.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 11.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

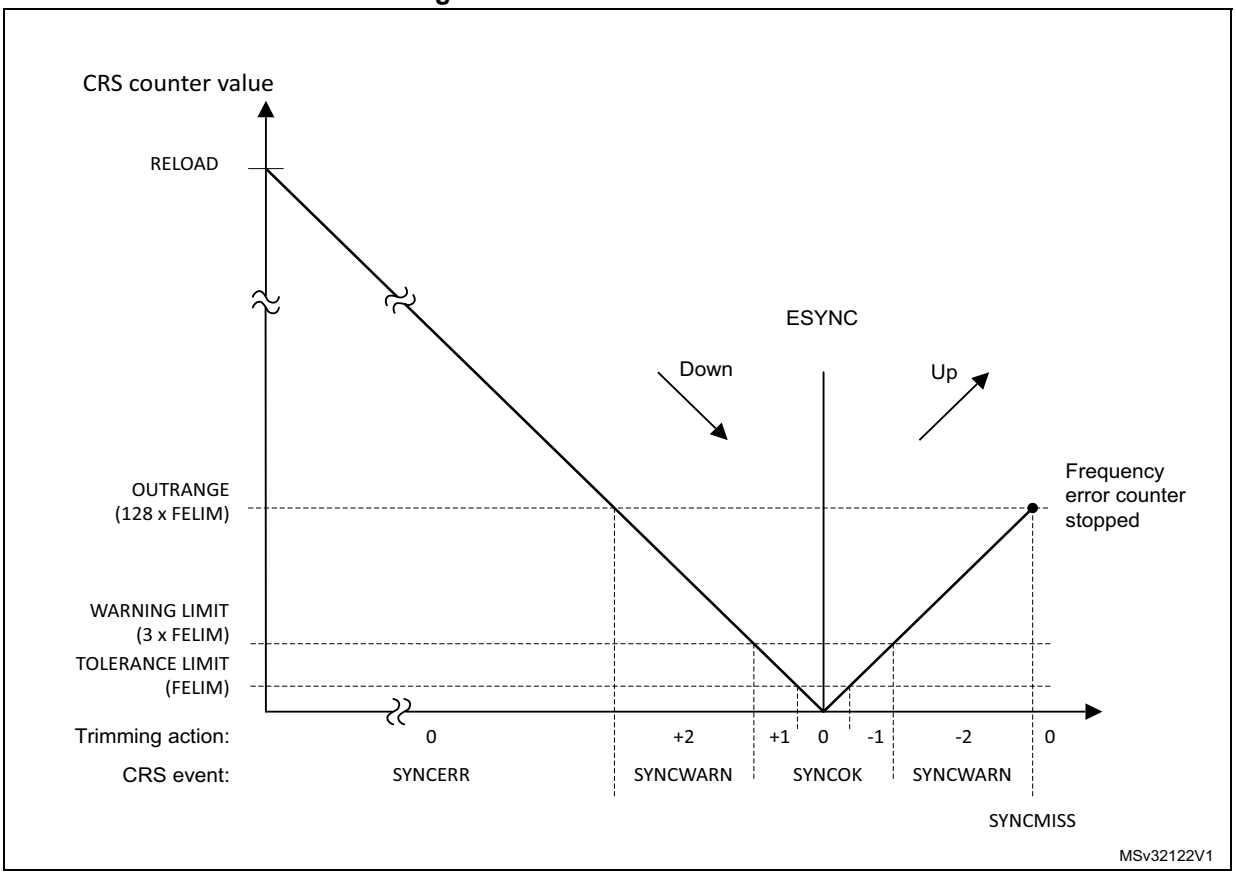
11.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 33. CRS counter behavior



11.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

11.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

11.5 CRS in low-power modes

Table 105. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.

11.6 CRS interrupts

Table 106. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

11.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

11.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 4000 (products supporting 7-bit TRIM width)
Reset value: 0x0000 2000 (other products)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRIM[6:0]							SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
	rw	rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TRIM[6:0]**: HSI48 oscillator smooth trimming

The default value of the HSI48 oscillator smooth trimming is 64, which corresponds to the middle of the trimming interval.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 11.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

11.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYNCPOL	Res.	SYNCSRC[1:0]		Res.	SYNCDIV[2:0]			FELIM[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.
0: SYNC active on rising edge (default)
1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 11.4.2: CRS internal signals*).
00: crs_sync_in_0 selected as SYNC signal source
01: crs_sync_in_1 selected as SYNC signal source
10: crs_sync_in_2 selected as SYNC signal source
11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.
000: SYNC not divided (default)
001: SYNC divided by 2
010: SYNC divided by 4
011: SYNC divided by 8
100: SYNC divided by 16
101: SYNC divided by 32
110: SYNC divided by 64
111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 11.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 11.4.4* for more details about counter behavior.

11.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 11.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

11.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNC WARNC	SYNC OKC
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCC**: Expected SYNC clear flag

Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag

Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag

Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag

Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

11.7.5 CRS register map

Table 107. CRS register map and reset values

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[6:0]							SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 4000 (products supporting 7-bit TRIM width) 0x0000 2000 (other products)																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

12 General-purpose I/Os (GPIO)

12.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), a secure configuration register (GPIOx_SECCFGR) and a high-speed low-voltage register (GPIOx_HSLVR).

12.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Configurable state of each I/O in Standby mode: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- TrustZone security support

12.3 GPIO functional description

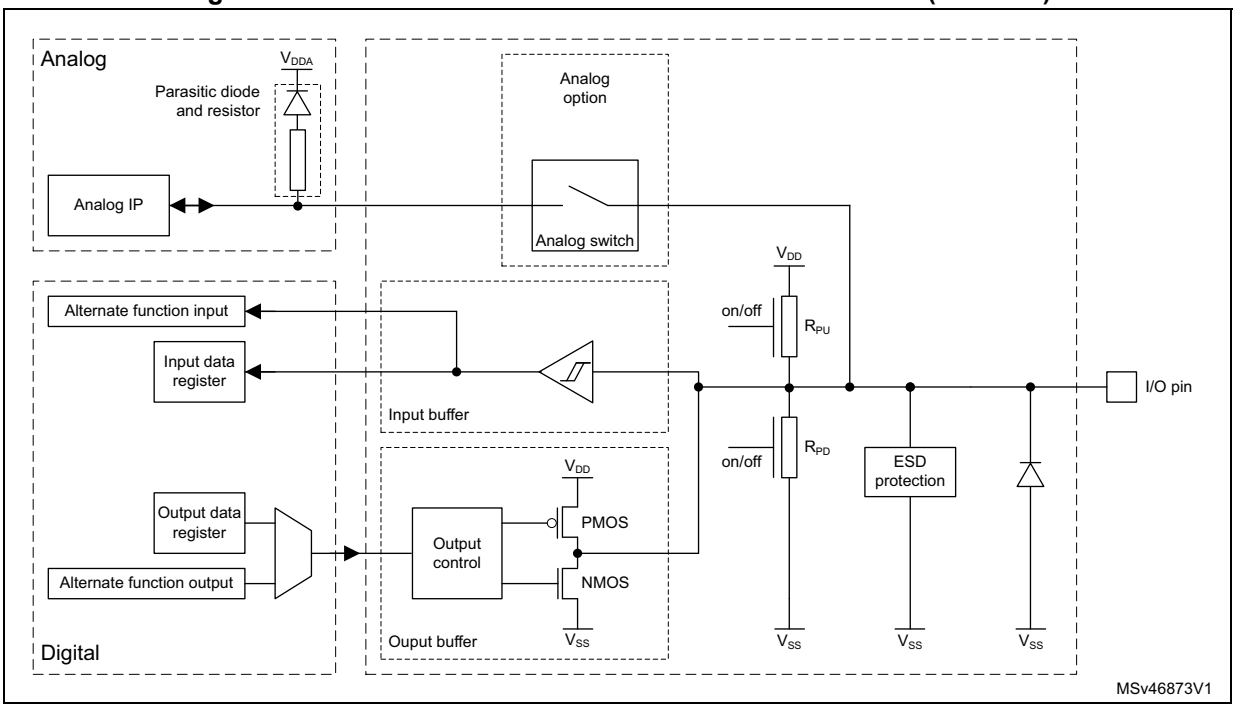
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words or bytes. GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

The figure below shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). *Table 108* gives the possible port bit configurations.

Figure 34. Structure of three-volt or five-volt tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

Table 108. Port bit configuration(1)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 108. Port bit configuration[1] (continued)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

12.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 12.3.15: Using BOOT0 pin as GPIO*).

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, that can be activated or not depending on the value in the GPIOx_PUPDR register.

12.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15):

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input or analog in GPIOx_MODER.
- **Alternate function**:
  - Connect the I/O to the desired AFx in one of GPIOx_AFRL or GPIOx_AFRH.
  - Select the type, pull-up/pull-down and output speed via GPIOx_OTYPER, GPIOx_PUPDR, and GPIOx_OSPEEDR respectively.
  - Configure the desired I/O as an alternate function in GPIOx_MODER.
  - The Cortex-M33 output EVENTOUT signal can be output as alternate function on several I/Os. An event can be signaled through the configured pin after executing SEV instruction.
  - The Cortex-M33 output EVENTOUT signal can be output as alternate function on several I/O pins. An event can be signaled through the configured pin after executing SEV instruction.
- **Additional functions**:
  - For the ADC, DAC, OPAMP, and COMP, configure the desired I/O in analog mode in GPIOx_MODER, and configure the required function in the ADC, DAC, OPAMP, and COMP registers.
  - For the additional functions like RTC, WKUPx, USB, and oscillators, configure the required function in the related RTC, PWR, USB, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

12.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os.

- GPIOx_MODER is used to select the I/O mode (input, output, AF, analog).
- GPIOx_OTYPER and GPIOx_OSPEEDR are used to select the output type (push-pull or open-drain) and speed.
- GPIOx_PUPDR is used to select the pull-up/pull-down whatever the I/O direction.

12.3.4 I/O port state in low-power modes

In Standby and Shutdown modes, the GPIO peripheral is not active, and its configuration is forced externally by the PWR control.

Before entering a deep low-power mode, fix the GPIO pin states, using the PWR_PUCRx and PWR_PDCRx registers. This prevents disturbing external components and buses, and optimizes the power consumption during deep low-power modes.

Exiting stop mode causes the power supply to be reset, the PWR_PUCRx and PWR_PDCRx registers to be reset, and the GPIO to be set to floating mode until reset by the application.

12.3.5 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

12.3.6 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

Two control bits in GPIOx_BSRR, BS[i] and BR[i], correspond to each bit in GPIOx_ODR. When written to 1, BS[i] sets the corresponding ODR[i] bit. When written to 1, BR[i] resets the ODR[i] corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using GPIOx_BSRR to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. GPIOx_BSRR provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

12.3.7 GPIO locking mechanism

GPIO control registers can be frozen by applying a specific write sequence to GPIOx_LCKR. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH and GPIOx_HSLVR.

To write GPIOx_LCKR , a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH, and GPIOx_HSLVR).

The LOCK sequence can only be performed using a word (32-bit long) access to GPIOx_LCKR due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

12.3.8 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

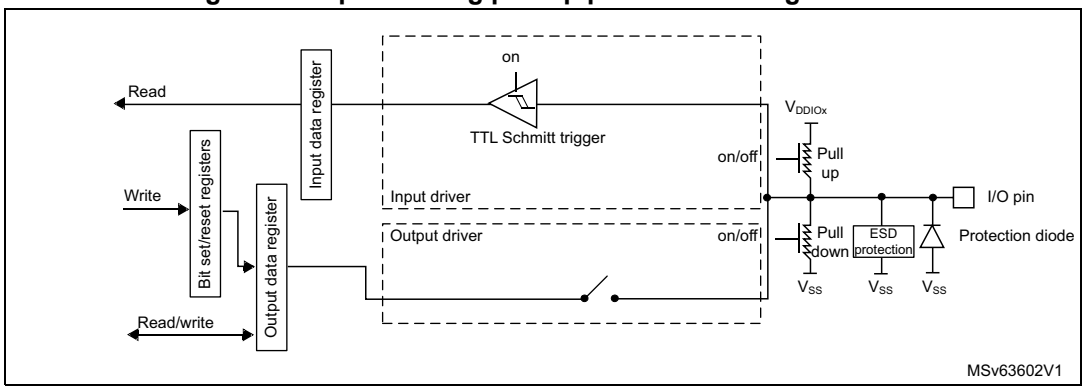
12.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

Figure 35. Input floating/pull-up/pull-down configurations



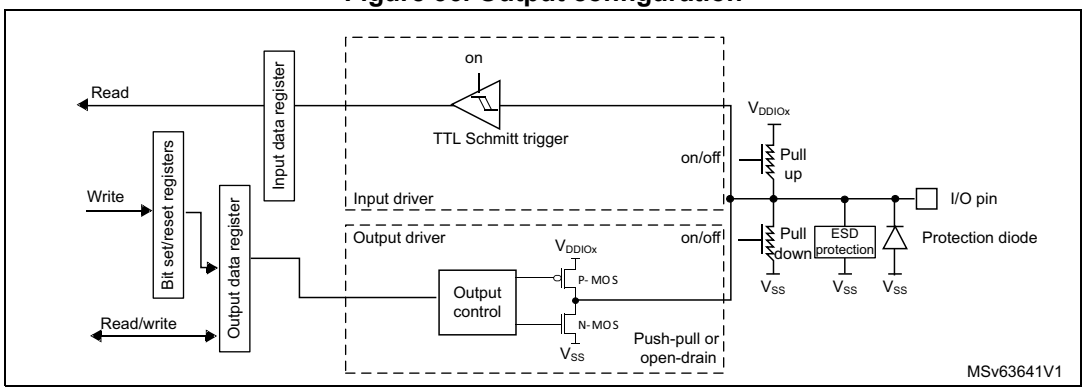
12.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 36. Output configuration



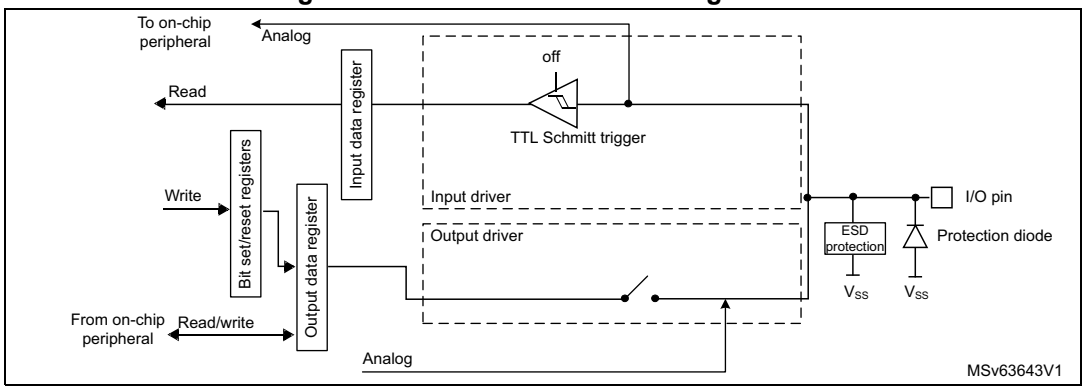
12.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 37. Alternate function configuration



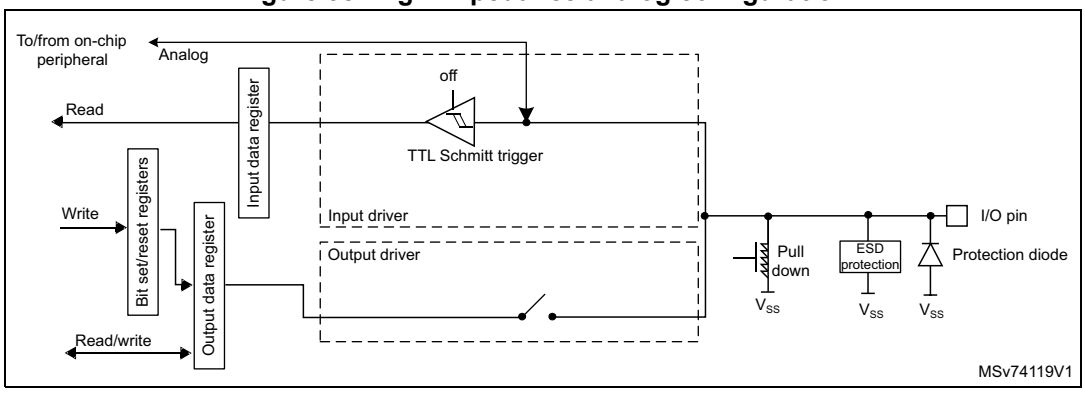
12.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up is disabled by hardware. The weak pull-down is configurable.
- Read access to the input data register gets the value 0.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 38. High-impedance analog configuration



12.3.13 Using HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

12.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 43.3: RTC functional description*.

12.3.15 Using BOOT0 pin as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the NSWBOOT0 bit in the user option byte, PH3 switches from the input mode to the analog input mode:

- After the option byte loading phase if NSWBOOT0 = 1.
- After reset if NSWBOOT0 = 0.

Note: On some packages, PH3 is not available and PB7 is used as the BOOT0 pin instead. Refer to the device datasheet for more information.

12.3.16 TrustZone security

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR. When the TrustZone is enabled (TZEN = 1), each I/O pin of GPIO port can be individually configured as secure through GPIOx_SECCFGR. This register can be written only by a secure access, a nonsecure write access is ignored. Both secure and nonsecure reads return the register data, that is the security configuration of individual I/O pins.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a nonsecure access. In case of nonsecure access, these bits are RAZ/WI. The GPIO clock and reset control bits in the RCC are automatically configured as secure as soon as at least one I/O in the GPIO is secure.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration (see *Section 5: Global TrustZone controller (GTZC)* for more details):

- For peripherals for which the I/O pin selection is done through alternate functions registers: if the peripheral is configured as secure, it cannot be connected to a nonsecure I/O pin. If this is not respected, the input data to the secure peripheral is forced to 0 (I/O input pin value is ignored) and the output pin value is forced to 0, thus avoiding any secure information leak through nonsecure I/Os.
- For I/Os with analog switches, directly controlled by peripherals (such as ADC for instance): If the I/O is secure, the I/O analog switch cannot be controlled by a nonsecure peripheral. If this is not respected, the switch remains open. This prevent the redirection of secure data to a nonsecure peripheral or I/O through analog path. Refer to *Table 9: Nonsecure peripherals not connectable to secure I/Os* for more details.
- Some of the paths between I/Os “additional functions” and peripherals are not blocked if the I/O is secure and the peripheral is nonsecure. Therefore it is recommended to configure those peripherals as secure even when not used by the application. Refer to *Table 10: Nonsecure peripherals connectable to secure I/Os* for the list of concerned peripherals. When the path has a security control, it follows the same rule as I/O selection through alternate functions.

Refer to the device pins definition table in datasheet for more information about peripherals alternate functions and additional functions mapping.

After reset, all GPIO ports are secure.

For the impacted registers, *Table 109* gives a summary of the secured bits following the individual security configuration bits in the GPIOx_SECCFGR register. When a secure configuration bit is set:

- Secured bits: read and write operations are only allowed by a secure access. Nonsecure read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- Nonsecure bits: no restriction. Read and write operations are allowed by both secure and nonsecure accesses.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR), all registers bits are nonsecure. The GPIOx_SECCFGR register is RAZ/WI.

Table 109. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Nonsecure access on secure bits
SECy = 1 in GPIOx_SECCFGR	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy and BRy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	BRy	GPIOx_BRR	
	AFSELy[3:0]	GPIOx_AFRH	
		GPIOx_AFRL	
	HSLVy	GPIOx_HSLVR	

12.3.17 Privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

12.3.18 High-speed low-voltage mode (HSLV)

Some I/Os have the capability to increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1).

Caution: The I/O HSLV configuration bit must not be set if the I/O supply (V_{DD} or V_{DDIO2}) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device. The I/O HSLV bit can be set only when the corresponding option bit is activated (IO_VDD_HSLV or IO_VDDIO2_HSLV depending on the I/O supply, refer to *Section 7: Embedded flash memory (FLASH)*). There is no hardware protection associated to this feature so it is recommended to use it only as a static configuration for fixed I/O supply.

Caution: Setting this bit when the I/O is configured in Fm+ mode is forbidden. An I/O is in Fm+ mode when it is configured as I2C alternate function, with FMP=1 in I2C_CR1 register. PB6, PB7, PB6, PB9 can also be in Fm+ mode when PB6_FMP, PB7_FMP, PB8_FMP, PB9_FMP, respectively, is set in SYSCFG_CFGR1 register.

12.3.19 I/O compensation cell

The I/O commutation slew rate (t_{fall}/t_{rise}) can be adapted by software depending on Process, Voltage and Temperatures conditions, in order to reduce the I/O noise on power supply. Refer to *Section 13: System configuration controller (SYSCFG)* for more details.

12.4 GPIO in low-power modes

Table 110. Effect of low-power modes on the GPIO

Mode	Description
Sleep	No effect. GPIO (EXTI) interrupts cause the device to exit Sleep mode.
Stop	No effect. GPIO (EXTI) interrupts cause the device to exit Stop mode.
Standby	The GPIO digital interface is powered down and must be reinitialized after exiting Standby mode. Wake-up pins can be configured to cause the device to exit Standby mode. I/Os are set to analog mode by hardware. Pull-up or pull-down device can individually be enabled through the PWR_PUCRx and PWR_PDCRx registers, respectively, to keep the I/Os at defined levels.
Shutdown	The GPIO digital interface is powered down and must be reinitialized after exiting Shutdown mode. Wake-up pins can be configured to cause the device to exit Shutdown mode. I/Os are set to analog mode by hardware. Pull-up or pull-down device can individually be enabled through the PWR_PUCRx and PWR_PDCRx registers, respectively, to keep the I/Os at defined levels during Shutdown mode. This setting is reset when exiting and the GPIOs become floating until reinitialized by the application.

12.5 GPIO interrupts

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output or alternate function mode (the port must not be configured in analog mode). Refer to *Section 17: Extended interrupts and event controller (EXTI)*.

12.6 GPIO registers

The peripheral registers can be written in word, half-word, or byte mode.

The number of GPIOx ports varies in STM32U3 series devices. Refer to the product datasheet for availability of a particular port. If not present, consider the associated bits as reserved and keep them at the reset value.

12.6.1 GPIO port mode register (GPIOx_MODER) (x = A:E, G, H)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C to E)

Reset value: 0xFFFF FFF0 (for port G)

Reset value: 0x0000 00CF (for port H)

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x I/O pin y mode configuration (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Note: This bitfield is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.2 GPIO port output type register (GPIOx_OTYPER) (x = A:E, G, H)

Address offset: 0x04

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x I/O pin y output type configuration (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

Note: This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A:E, G, H)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x I/O pin y output speed configuration (y = 15 to 0)
These bits are written by software to configure the I/O output speed.
00: Low speed
01: Medium speed
10: High speed
11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications, and the power supply and load conditions for each speed.
This bitfield is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A:E, G, H)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x I/O pin y pull-up/pull-down configuration (y = 15 to 0)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up or pull-down
01: Pull-up
10: Pull-down
11: Reserved

Note: This bitfield is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.5 GPIO port input data register (GPIOx_IDR) (x = A:E, G, H)

Address offset: 0x10

Reset value: 0x0000 XXXX

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x I/O pin y input data (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.6 GPIO port output data register (GPIOx_ODR) (x = A:E, G, H)

Address offset: 0x14

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port x I/O pin y output data (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, these bits can be individually set and/or reset by writing to GPIOx_BSRR or GPIOx_BRR.

This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A:E, G, H)

Address offset: 0x18

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x I/O pin y output reset (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x I/O pin y output set (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A:E, G, H)

Address offset: 0x1C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO.

During the write sequence, LCKR[15:0] value must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Each lock bit freezes corresponding configuration bits in configuration and alternate function registers.

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

Note: A specific write sequence is used to write to this GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified by using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

– LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

– LOCK key read

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the lock key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port x I/O pin y lock (y = 15 to 0)

These bits are read/write but can only be written when LCKK = 0.

0: Port configuration not locked

1: Port configuration locked

Note: This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.9 GPIO alternate function low register (GPIOx_AFRL) (x = A:E, G, H)

Address offset: 0x20

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Port x I/O pin y alternate function selection for (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: This bitfield is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.10 GPIO alternate function high register (GPIOx_AFRH) (x = A:E, G, H)

Address offset: 0x24

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Port x I/O pin y alternate function selection (y = 15 to 8)
These bits are written by the software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: This bitfield is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.11 GPIO port bit reset register (GPIOx_BRR) (x = A:E, G, H)

Address offset: 0x28

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x I/O pin y output reset (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Note: This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A:E, G, H)

Address offset: 0x2C

Reset value: 0x0000 0000

Individual bits of this register are protected against nonsecure access when corresponding SECy = 1 in GPIO_SECCFGR and the system is secure (TZEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSLV15	HSLV14	HSLV13	HSLV12	HSLV11	HSLV10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HSLVy:** Port x I/O pin y high-speed low-voltage configuration (y = 15 to 0)

These bits are written by software to optimize the I/O speed when the I/O supply is low.

Each bit is active only if the corresponding IO_VDD_HSLV/IO_VDDIO2_HSLV user option bit is set. It must be used only if the I/O supply voltage is below 2.7 V.

Setting these bits when the I/O supply (V_{DD} or V_{DDIO2}) is higher than 2.7 V may be destructive. Setting this bit when the I/O is configured in Fm+ mode is forbidden.

0: I/O speed optimization disabled

1: I/O speed optimization enabled

Note: Not all I/Os support the HSLV mode. Refer to the I/O structure in the corresponding datasheet for the list of I/Os supporting this feature. HSLV configuration for the other I/Os must be kept at reset value.

This bit is reserved and must be kept at reset value when the corresponding I/O is not available on the selected package.

12.6.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A:E, G, H)

Address offset: 0x30

Reset value: 0x0000 FFFF (for ports A to E)

Reset value: 0x0000 FFFC (for port G)

Reset value: 0x0000 000B (for port H)

When the system is secure (TZEN = 1), this register is used to configure selected I/O pin as secure. It can be written only by a secure access. A nonsecure write access to this register is ignored. Both nonsecure and secure reads return the register data.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: Port x I/O pin y security configuration (y = 15 to 0)
These bits are written by software to enable or disable the I/O port pin security.
0: The I/O pin is nonsecure
1: The I/O pin is secure. Refer to *Table 109: GPIO secured bits* for all corresponding secured bits.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

12.6.14 GPIO register map

Table 111. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOx_MODER (x = A:E, G, H)	MODE15 [1:0]		MODE14 [1:0]		MODE13 [1:0]		MODE12 [1:0]		MODE11 [1:0]		MODE10 [1:0]		MODE9 [1:0]		MODE8 [1:0]		MODE7 [1:0]		MODE6 [1:0]		MODE5 [1:0]		MODE4 [1:0]		MODE3 [1:0]		MODE2 [1:0]		MODE1 [1:0]		MODE0 [1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C to E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port G	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	Reset value for port H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1
0x04	GPIOx_OTYPER (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 111. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	GPIOx_OSPEEDR (x = A:E, G, H)	OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]		OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (x = A:E, G, H)	PUPD15 [1:0]		PUPD14 [1:0]		PUPD13 [1:0]		PUPD12 [1:0]		PUPD11 [1:0]		PUPD10 [1:0]		PUPD9 [1:0]		PUPD8 [1:0]		PUPD7 [1:0]		PUPD6 [1:0]		PUPD5 [1:0]		PUPD4 [1:0]		PUPD3 [1:0]		PUPD2 [1:0]		PUPD1 [1:0]		PUPD0 [1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	GPIOx_ODR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (x = A:E, G, H)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (x = A:E, G, H)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (x = A:E, G, H)	AFSEL15 [3:0]				AFSEL14 [3:0]				AFSEL13 [3:0]				AFSEL12 [3:0]				AFSEL11[3:0]				AFSEL10 [3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	GPIOx_HSLVR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV15	HSLV14	HSLV13	HSLV12	HSLV11	HSLV10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	GPIOx_SECCFGR (x = A:E, G, H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value for A to E																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port G																	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port H																	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

13 System configuration controller (SYSCFG)

13.1 SYSCFG main features

The STM32U3 series devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Managing robustness feature
- Configuring FPU interrupts
- Enabling/disabling the FMP high-drive mode of some I/Os and voltage booster for I/Os analog switches
- Managing the I/O compensation cell
- Configuring register security access

13.2 SYSCFG functional description

13.2.1 I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the current slew-rate and output impedance in the I/O buffer. Two compensation cells are embedded, one for the I/Os supplied by V_{DD} and one for the I/Os supplied by V_{DDIO2}.

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os. The HSI is used by the compensation cells and must be enabled before enabling the compensation cells in SYSCFG_CCCSR.

When enabled, the compensation cell tracks the PVT, and the 8-bit code PCVx and NCVx (x = 1 for I/Os supplied by V_{DD}, x = 2 for I/Os supplied by V_{DDIO2}) are available in SYSCFG_CCVR once the RDYx is set. If the CSx bit is cleared, the I/Ox receives the code from SYSCFG_CCVR, resulting from the compensation cell.

To optimize the trimming, the code can be adjusted through SYSCFG_CCCR. Two sets of bits are available:

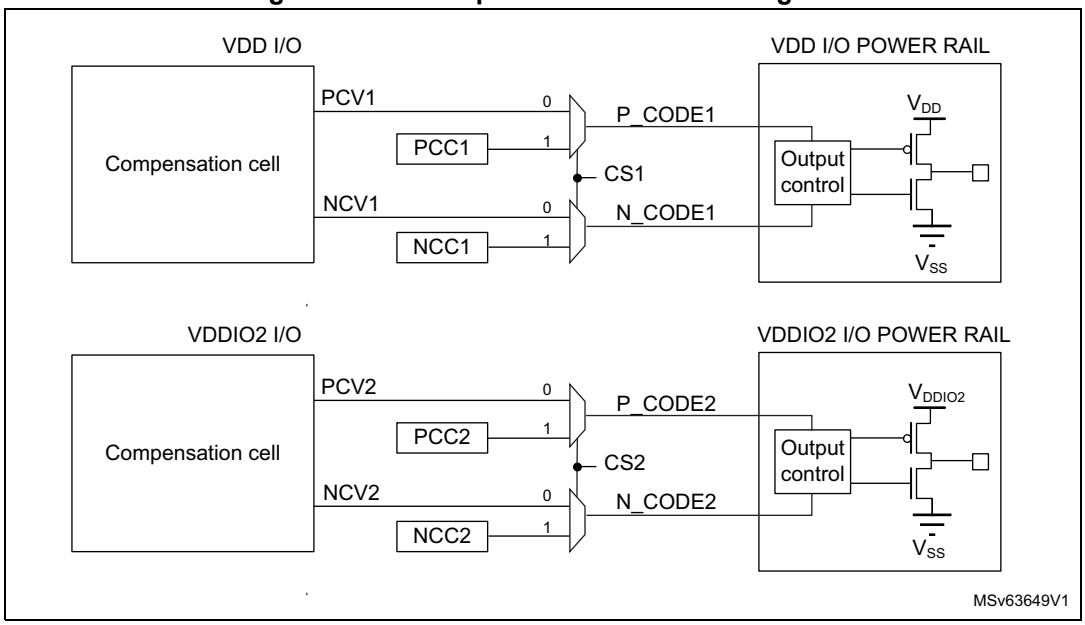
- PCC1/NCC1 for the V_{DD} power rail
- PCC2/NCC2 for the V_{DDIO2} power rail

They can be selected independently through CS1 and CS2 bits in SYSCFG_CCCSR (see *Figure 39*).

To reduce the power consumption, it is recommended to copy the code from SYSCFG_CCVR to SYSCFG_CCCR. After the result is ready, set the CSx bit and disable the compensation cell.

Note: The compensation cell can be used only when 1.6 V ≤ V_{DDIOx} ≤ 3.6 V.

Figure 39. I/O compensation cell block diagram



13.2.2 SYSCFG TrustZone security and privilege

SYSCFG TrustZone security

When the TrustZone security is activated, the SYSCFG is able to secure registers from being modified by nonsecure accesses.

The TrustZone security is activated by the TZEN option bit in FLASH_OPTR.

A nonsecure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SYSCFG illegal access event is enabled in the GTZC.

Privileged/unprivileged mode

SYSCFG registers can be read and written by privileged and unprivileged accesses except the SYSCFG registers for CPU configuration: SYSCFG_CSLCKR, SYSCFG_FPUIMR, and SYSCFG_CNSLCKR registers, and the FPUSEC bit in SYSCFG_SECCFGR.

An unprivileged access to a privileged register is RAZ/WI.

The table below shows the register security overview.

Table 112. TrustZone security and privilege register accesses

SYSCFG register	Read/write access		Privileged /unprivileged access
TrustZone configuration(1)	TZEN = 1	TZEN = 0	Not applicable
SYSCFG_SECCFGR	Read: no restriction Write: secure access only nonsecure write is WI and generates an illegal access event.	RAZ/WI	Read: no restriction FPUSEC privileged write only Other bits write: no restriction
SYSCFG_CSLCKR	Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event.	RAZ/WI	Privileged only Unprivileged: RAZ/WI
SYSCFG_FPUIMR	If FPUSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event. If FPUSEC = 0: Read/Write: no restriction	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CNSLCKR	Read/write: no restriction	No restriction	Privileged only Unprivileged: RAZ/WI
SYSCFG_CFGR1	Read/Write: secure access only for secure bits depending on peripheral security bits in GTZC Nonsecure access only for nonsecure bits, otherwise RAZ/WI	No restriction	No restriction
SYSCFG_CFGR2	If CLASSBSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event. If CLASSBSEC = 0:Read/Write: no restriction	No restriction	No restriction
SYSCFG_CCCSR SYSCFG_CCVR SYSCFG_CCCR	If SYSCFGSEC = 1: Read/Write: secure access only Nonsecure access is RAZ/WI and generates an illegal access event If SYSCFGSEC = 0: Read/Write: no restriction	No restriction	No restriction
SYSCFG_RSSCMDR	RAZ/WI if register access is not allowed(2)	RAZ/WI	No restriction

1. TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.
2. Refer to register description for register access.

13.2.3 GPIO analog switches supply selection

Two bits from the SYSCFG_CFGR1 register are used to select the power supply of the I/O analog switches (used analog input signal paths): BOOSTEN and ANASWVDD.

They have to be initialized according to the voltage of the V_{DD} and V_{DDA} power supplies. When V_{DDA} voltage is larger than 2.4 V, the I/O analog switches must be powered by V_{DDA}.

When V_{DDA} voltage is lower than 2.4 V and V_{DD} is higher than 2.4 V, the I/O analog switches must be powered by V_{DD}. When both V_{DD} and V_{DDA} voltages are lower than 2.4 V, the I/O analog switches must be powered by the output of the V_{DD} booster. When I/O analog switches are supplied by V_{DDA} or V_{DD} booster, the ASV bit of the PWR_SVMCR register must be set.

Table 113. BOOSTEN and ANASWVDD set/reset

VDD	VDDA	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	< 2.4 V		1
< 2.4 V		1	0

13.3 SYSCFG registers

13.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register provides write access security and can be written only when the access is secure. This register can be globally write-protected. Each bit can be individually write-protected. A nonsecure write access is WI and generates an illegal access event. There are no read restrictions.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

This register can be read and written by privileged and unprivileged access, except for FPUSEC that can be written only with privileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	Res.	CLASSBSEC	SYSCFGSEC
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FPUSEC**: FPU security
0: SYSCFG_FPUIMR can be read and written by secure and nonsecure access.
1: SYSCFG_FPUIMR can be read and written by secure access only.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: Class B security
0: SYSCFG_CFGR2 can be read and written by secure and nonsecure access.
1: SYSCFG_CFGR2 can be read and written by secure access only.

Bit 0 **SYSCFGSEC**: Security of SYSCFG clock control, memory erase status, and compensation cell registers

0: SYSCFG configuration clock in RCC registers and SYSCFG_CCCSR, SYSCFG_CCVR, and SYSCFG_CCCR can be read and written by secure and nonsecure access.

1: SYSCFG configuration clock in RCC registers and SYSCFG_CCCSR, SYSCFG_CCVR, and SYSCFG_CCCR can be read and written by secure access only.

13.3.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

Address offset: 0x04

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be a mix of secure and nonsecure bits depending on the ADC security configuration bit in GTZC peripheral and GPIO security bits. A nonsecure read/write access on secured bits is RAZ/WI.

When the system is not secure (TZEN = 0), there is no access restriction.

This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB9_FMP	PB8_FMP	PB7_FMP	PB6_FMP
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	IR_MOD[1:0]		IR_POL	Res.	Res.	Res.	Res.	Res.
						rw	rw	rw	rw	rw					

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **PBi_FMP**: Fast-mode Plus driving capability activation on PBi (i = 9 to 6)

This bit can be read and written only with secure access if PB9 is secure in GPIOB. This bit enables the Fm+ driving mode for PBi when PBi is not used by the I2C peripheral. This can be used to drive a LED for instance.

0: PBi pin operates in standard mode.

1: Fm+ mode is enabled on PBi pin and the speed control is bypassed.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ANASWVDD**: GPIO analog switch control voltage selection

This bit can be read and written only with secure access if ADC1 or ADC2 is secure in GTZC.

0: If booster if OFF: I/O analog switches are supplied by V_{DDA} if ASV = 1 in PWR_SVMCR, or by V_{DD} if ASV = 0.
If booster is ON: I/O analog switches are supplied by booster.

1: I/O analog switches are supplied by V_{DD}.

Note: Refer to Table 113: BOOSTEN and ANASWVDD set/reset

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable

This bit is only effective when ASV = 1 in PWR_SVMR, and ANASWVDD = 0.

This bit can be read and written only with secure access if ADC1 or ADC2 is secure in GTZC.

0: I/O analog switches are supplied by V_{DDA} voltage.

1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V_{DD}).

Note: Refer to Table 113: BOOSTEN and ANASWVDD set/reset

Bits 7:6 **IR_MOD[1:0]**: IR modulation envelope signal selection
This bitfield can be read and written only with secure access if TIM17 is secure in GTZC.
It selects the signal for IR modulation envelope.
00: TIM16
01: USART1
10: UART4
11: Reserved
Note: Refer to Table 113: BOOSTEN and ANASWVDD set/reset.

Bit 5 **IR_POL**: IR output polarity selection
This bit can be read and written only with secure access if TIM17 is secure in GTZC.
0: Output of IRTIM (IR_OUT) is not inverted.
1: Output of IRTIM (IR_OUT) is inverted.
Note: Refer to Table 113: BOOSTEN and ANASWVDD set/reset.

Bits 4:0 Reserved, must be kept at reset value.

13.3.3 SYSCFG FPU interrupt mask register (SYSCFG_FPUIMR)

Address offset: 0x08

Reset value: 0x0000 001F

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the FPUSEC bit in the SYSCFG_SECCFGR register: a nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction. This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IXIE	FPU_IDIE	FPU_OFIE	FPU_UFIE	FPU_DZIE	FPU_IOIE
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **FPU_IXIE**: Floating point unit interrupt enable bit - inexact
FPU inexact interrupt enable bit (interrupt disable at reset)
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **FPU_IDIE**: Floating point unit interrupt enable bit - input denormal
FPU denormal interrupt enable bit
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **FPU_OFIE**: Floating point unit interrupt enable bit - overflow
FPU overflow interrupt enable bit
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **FPU_UFIE**: Floating point unit interrupt enable bit - underflow
FPU underflow interrupt enable bit
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **FPU_DZIE**: Floating point unit interrupt enable bit - divide-by-zero
FPU divide-by-zero interrupt enable bit
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **FPU_IOIE**: Floating point unit interrupt enable bit - invalid operation
FPU invalid operation interrupt enable bit
0: Interrupt disabled
1: Interrupt enabled

13.3.4 SYSCFG CPU nonsecure lock register (SYSCFG_CNSLCKR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is used to lock the configuration of nonsecure MPU and VTOR_NS registers. This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LOCKNSMPU**: Nonsecure MPU registers lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to nonsecure MPU_CTRL_NS, MPU_RNR_NS, and MPU_RBAR_NS registers.
0: Nonsecure MPU register write enabled
1: Nonsecure MPU register write disabled

Bit 0 **LOCKNSVTOR**: VTOR_NS register lock
This bit is set by software and cleared only by a system reset.
0: VTOR_NS register write enabled
1: VTOR_NS register write disabled

13.3.5 SYSCFG CPU secure lock register (SYSCFG_CSLCKR)

Address offset: 0x10

Reset value: 0x0000 0000

This register is used to lock the configuration of PRIS and BFHFNMINS in AIRCR register, SAU, secure MPU, and VTOR_S registers.

When the system is secure (TZEN = 1), this register can be written only when the access is secure. A nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), this register is RAZ/WI. This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
													rs	rs	rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKSAU**: SAU register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR, and SAU_RLAR registers.

0: SAU register write enabled

1: SAU register write disabled

Bit 1 **LOCKSMPU**: Secure MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR, and MPU_RBAR registers.

0: Secure MPU register writes enabled

1: Secure MPU register writes disabled

Bit 0 **LOCKSVTAIRCR**: VTOR_S register and AIRCR register bits lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.

0: VTOR_S register PRIS and BFHFNMINS bits in AIRCR register write enabled

1: VTOR_S register PRIS and BFHFNMINS bits in AIRCR register write disabled

13.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x14

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the CLASSBSEC bit in the SYSCFG_SECCFGR register.

When CLASSBSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction. This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
												rs	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCL**: ECC lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the FLASH ECC double error signal connection to the TIM1/15/16/17 break input.

0: ECC double error disconnected from TIM1/15/16/17 break input

1: ECC double error connected to TIM1/15/16/17 break input

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/15/16/17 break input, as well as the PVDE and PVDLS[2:0] in the PWR register.

0: PVD interrupt disconnected from TIM1/15/16/17 break input. PVDE and PVDLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/15/16/17 break input. PVDE and PVDLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity bit

This bit is set by the software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/15/16/17 break inputs.

0: SRAM double error disconnected from TIM1/15/16/17 break inputs

1: SRAM double error connected to TIM1/15/16/17 break inputs

Bit 0 **CLL**: Cortex-M33 LOCKUP (HardFault) output enable

This bit is set by the software and cleared only by a system reset. It can be used to enable and lock the connection of Cortex-M33 LOCKUP (HardFault) output to TIM1/15/16/17 break input.

0: Cortex-M33 LOCKUP output disconnected from TIM1/15/16/17 break inputs

1: Cortex-M33 LOCKUP output connected to TIM1/15/16/17 break inputs

13.3.7 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

Address offset: 0x1C

Reset value: 0x0000 000A

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction. This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
						r	r					rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **RDY2:** VDDIO2 I/O compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DDIO2}.

0: VDDIO2 I/O compensation cell not ready

1: VDDIO2 I/O compensation cell ready

*Note: The HSI clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY2*) is not set if the HSI clock is not enabled.*

Bit 8 **RDY1:** VDD I/O compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DD}.

0: VDD I/O compensation cell not ready

1: VDD I/O compensation cell ready

*Note: The HSI clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY1*) is not set if the HSI clock is not enabled.*

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CS2:** VDDIO2 I/O code selection

This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DDIO2}.

0: VDDIO2 I/O code from the cell (available in SYSCFG_CCVR)

1: VDDIO2 I/O code from SYSCFG_CCCR

Bit 2 **EN2:** VDDIO2 I/O compensation cell enable

This bit enables the compensation cell of the I/Os supplied by V_{DDIO2}.

0: VDDIO2 I/O compensation cell disabled

1: VDDIO2 I/O compensation cell enabled

Bit 1 **CS1:** VDD I/O code selection

This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DD}.

0: VDD I/O code from the cell (available in the SYSCFG_CCVR)

1: VDD I/O code from the SYSCFG compensation cell code register (SYSCFG_CCCR)

Bit 0 **EN1:** VDD I/O compensation cell enable
This bit enables the compensation cell of the I/Os supplied by V_{DD}.
0: VDD I/O compensation cell disabled
1: VDD I/O compensation cell enabled

13.3.8 SYSCFG compensation cell value register (SYSCFG_CCVR)

Address offset: 0x20

Reset value: 0x0000 0000

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When the SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction. This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCV2[3:0]				NCV2[3:0]				PCV1[3:0]				NCV1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PCV2[3:0]:** PMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS2 bit in SYSCFG_CCCSR is reset.

Bits 11:8 **NCV2[3:0]:** NMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS2 bit in SYSCFG_CCCSR is reset.

Bits 7:4 **PCV1[3:0]:** PMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS1 bit in SYSCFG_CCCSR is reset.

Bits 3:0 **NCV1[3:0]:** NMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS1 bit in SYSCFG_CCCSR is reset.

13.3.9 SYSCFG compensation cell code register (SYSCFG_CCCR)

Address offset: 0x24

Reset value: 0x0000 7878

When the system is secure (TZEN = 1), this register can be protected against nonsecure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When SYSCFGSEC bit is set, only secure access is allowed: nonsecure read/write access is RAZ/WI and generates an illegal access event.

When the system is not secure (TZEN = 0), there is no access restriction. This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCC2[3:0]				NCC2[3:0]				PCC1[3:0]				NCC1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PCC2[3:0]:** PMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS2 bit is set in SYSCFG_CCCSR.

Bits 11:8 **NCC2[3:0]:** NMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS2 bit is set in SYSCFG_CCCSR.

Bits 7:4 **PCC1[3:0]:** PMOS compensation code of the I/Os supplied by V_{DD}
These bits are written by software to define an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when CS1 bit is set in SYSCFG_CCCSR.

Bits 3:0 **NCC1[3:0]:** NMOS compensation code of the I/Os supplied by V_{DD}
These bits are written by software to define an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when CS1 bit is set in SYSCFG_CCCSR.

13.3.10 SYSCFG RSS command register (SYSCFG_RSSCMDR)

Address offset: 0x2C

Power-on reset value: 0x0000 0000

System reset value: 0xUUUU UUUU (not affected)

When the system is secure (TZEN = 1), this register can be read and written only when the APB access is secure. Otherwise it is RAZ/WI.

When the system is not secure (TZEN = 0), this register is RAZ/WI. This register can be read and written by privileged and unprivileged access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSSCMD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RSSCMD[15:0]**: RSS commands

This bitfield defines a command to be executed by the RSS.

13.3.11 SYSCFG register map

Table 114. SYSCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSECC	Res.	CLASSBSEC	SYSCFGSEC
	Reset value																													0		0	0
0x04	SYSCFG_CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB9_FMP	PB8_FMP	PB7_FMP	PB6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	BOOSTEN	IR_MOD[1:0]		IR_POL	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0							0	0	0	0	0					
0x08	SYSCFG_FPUIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IXIE	FPU_IDIE	FPU_OFIE	FPU_UFIE	FPU_DZIE	FPU_IOIE
	Reset value																											0	1	1	1	1	1
0x0C	SYSCFG_CNSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
	Reset value																															0	0

Table 114. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	SYSCFG_CSLCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
	Reset value																														0	0	0
0x14	SYSCFG_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL.
	Reset value																													0	0	0	0
0x18	Reserved	Reserved																															
0x1C	SYSCFG_CCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
	Reset value																							0	0					1	0	1	0
0x20	SYSCFG_CCVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV2[3:0]				NCV2[3:0]				PCV1[3:0]				NCV1[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	SYSCFG_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC2[3:0]				NCC2[3:0]				PCC1[3:0]				NCC1[3:0]			
	Reset value																	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
0x28	Reserved	Reserved																															
0x2C	SYSCFG_RSSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Power-on reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

14 Peripherals interconnect matrix

14.1 Interconnect matrix introduction

Several peripherals have direct connections between them. This allows autonomous communication and or synchronization between peripherals, saving CPU resources, thus power supply consumption.

In addition, these hardware connections remove software latency and allow design of predictable system.

Depending on peripherals, these interconnections can operate in various power modes: Run, Sleep, Stop 0, Stop 1, Stop 2, and Stop 3 modes.

14.2 Connection summary

Table 115. Peripherals interconnect matrix

Source	Destination																								
	TIM1	TIM2	TIM3	TIM4	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	ADC1/2	ADF1	OPAMP1/2	DAC1	COMP1/2	GPDMA	IRTIM	U(S)ARTs	LPUART1	I2Cs	I3Cs	SPIs	TAMP	RTC	AES/SAES
TIM1	-	*1*	*1*	*1*	-	-	*1*	-	-	-	-	*2*	-	-	*4*	*8*	-	-	-	-	-	-	-	-	-	-
TIM2	*1*	-	*1*	*1*	-	-	*1*	-	-	-	-	*2*	-	-	*4*	*8*	*15*	-	-	-	-	-	-	-	-	-
TIM3	*1*	*1*	-	*1*	-	-	*1*	-	-	-	-	*2*	-	-	-	*8*	*15*	-	-	-	-	-	-	-	-	-
TIM4	*1*	*1*	*1*	-	-	-	*1*	-	-	-	-	*2*	-	-	*4*	-	*15*	-	-	-	-	-	-	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	*4*	-	-	-	-	-	-	-	-	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*4*	-	-	-	-	-	-	-	-	-	-	-
TIM15	*1*	*1*	*1*	*1*	-	-	-	-	-	-	-	*2*	-	-	*4*	*8*	*15*	-	-	-	-	*14*	-	-	-	-
TIM16	*1*	*1*	*1*	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	*14*	-	-	-	-
TIM17	*1*	*1*	*1*	*1*	-	-	*1*	-	-	-	-	-	-	-	-	-	-	*13*	-	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	*4*	-	*15*	-	*14*	*14*	*14*	*14*	*14*	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	-	-	*15*	-	*14*	-	*14*	*14*	*14*	-	-	-
LPTIM3	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	*4*	-	*15*	-	-	*14*	*14*	-	*14*	-	-	-
LPTIM4	-	-	-	-	-	-	-	-	-	-	-	*2*	-	-	-	-	*15*	-	-	-	-	-	-	-	-	-
ADC1/2	*3*	-	*3*	-	-	-	-	-	-	-	-	-	*23*	-	-	-	*15*	-	-	-	-	-	-	-	-	-

Table 115. Peripherals interconnect matrix (continued)

Source	Destination																									
	TIM1	TIM2	TIM3	TIM4	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	ADC1/2	ADF1	OPAMP1/2	DAC1	COMP1/2	GPDMA	IRTIM	U(S)ARTs	LPUART1	I2Cs	I3Cs	SPIs	TAMP	RTC	AES/SAES
ADF1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1	-	-	-	-	-	-	-	-	-	-	-	-	-	*16*	-	*16*	-	-	-	-	-	-	-	-	-	-
COMP1/2	*11*	*11*	*11*	*11*	-	-	*11*	*11*	*11*	*7*	*7*	-	-	-	-	-	*15*	-	*14*	*14*	*14*	*14*	*14*	-	-	-
GPDMA1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*15*	-	*14*	-	*14*	*14*	*14*	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	*7*	*7*	*2*	*5*	-	*4*	-	*15*	-	*14*	*14*	*14*	*14*	*14*	-	-	-
RTC wake-up	-	-	-	-	-	-	-	*9*	*9*	-	-	-	-	-	-	-	*15*	-	*14*	*14*	*14*	*14*	*14*	-	-	-
RTC Alarm	-	-	-	-	-	-	-	-	-	*7*	*7*	-	-	-	-	-	*15*	-	*14*	*14*	*14*	*14*	*14*	-	-	-
TAMP	-	-	-	-	-	-	-	-	-	*7*	*7*	-	-	-	-	-	*15*	-	-	-	-		-	-	*19*	*20*
HSE	-	-	-	-	-	-	-	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
LSE	-	*6*	-	-	-	-	*6*	*6*	*6*	*6*	-	-	-	-	*17*	-	-	-	-	-	-		-	-	-	-
CSS in LSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	*18*	-	-
MSIS/MSIK	*6*	*6*	*6*	*6*	-	-	-	*6*	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
HSI	*6*	*6*	*6*	*6*	-	-	-	*6*	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
LSI	-	-	-	-	-	-	-	*6*	*6*	*6*	-	-	-	-	*17*	-	-	-	-	-	-		-	-	-	-
MCO	-	-	-	-	-	-	-	*6*	*6*	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
V_{CORE}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
V_{REFINT}	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	*16*	-	-	-	-	-		-	-	-	-
Temp. sensor	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	-	-	-	-	-	-		-	-	-	-
V_{BAT}	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	-	-	-	-	-	-		-	-	-	-
OPAMP1/2	-	-	-	-	-	-	-	-	-	-	-	*16*	-	-	-	-	-	-	-	-	-		-	-	-	-
System errors	*12*	-	-	-	-	-	*12*	*12*	*12*	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-
USB	-	*10-*	-	-	-	-	-	-	-	-	-	-	-	--		-	-	-	-	-	-	-	-	-	-	-
System flash	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	*20*
AES/SAES	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	*20*
I3Cs	-	-	-	-	-	-	*21*	*21*	*21*	*21*	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-

14.3 Interconnection details

14.3.1 Master to slave interconnection for timers

From timer (TIM1/2/3/4/15/16/17) to timer (TIM1/2/3/4/15).

Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

A description of the feature is provided in *Section 35.3.32: ADC synchronization*.

The synchronization modes are detailed in:

- *Section 35.3.32: ADC synchronization*
- *Section 36.4.22: Timers and external trigger synchronization*
- *Section 38.4.23: External trigger synchronization (TIM15 only)*

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. It can be also from signals tim16_oc1 and tim17_oc1 in case of TIM16/17. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The possible master/slave connections are given in:

- *Table 347: Internal trigger connection*
- *Table 371: TIMx internal trigger connection*
- *Table 393: TIMx internal trigger connection*

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 360: Effect of low-power modes on TIM1*
- *Table 379: Effect of low-power modes on TIM2/TIM3/TIM4*
- *Table 400: Effect of low-power modes on TIM15/TIM16/TIM17*

14.3.2 Triggers to ADCs

From EXTI, timers (TIM1/2/3/4/6/15) and LP timers (LPTIM1/2/3/4) to ADC1/2

Purpose

A conversion, or a sequence of conversions, can be triggered either by software or by an external event (such as timer capture or input pins). For ADC1/2, if the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events can trigger a conversion with the selected polarity.

More details in:

- *Section 22.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR1)*

- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/3/4), basic timer (TIM6), advanced-control timers (TIM1), and general-purpose timer (TIM15) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1/ 2/3/4) can be used to generate the ADC triggering event through the LPTIM channels (TIMx synchronization described in *Section 35.3.32: ADC synchronization*) in addition to the EXTI on channels 11 and 15.

Triggering signals

For ADC1/2, the input triggering signals and the description of the interconnection between ADC1/2, and timers, are given in:

- adc_ext_trgi: *Table 178: ADC1/2 interconnection*
- adc_jext_trgi: *Table 178: ADC1/2 interconnection*
- *Section 22.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*
- *Section 22.4.25: Timing diagram examples (single/continuous modes, hardware/software triggers)*)

Active power mode

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active in Run and Sleep modes only. The effects of low-power modes are given in:

- *Table 360: Effect of low-power modes on TIM1*
- *Table 400: Effect of low-power modes on TIM15/TIM16/TIM17*
- *Table 406: STM32U3 series LPTIM features*
- *Table 422: Effect of low-power modes on the LPTIM*

14.3.3 ADC analog watchdogs as triggers to timers

From ADC1/2 to TIM1/3

Purpose

The internal analog watchdog output signals coming from ADC1/2, are connected to on-chip timers. ADC1/2 can provide trigger event through analog watchdog signals to timers (TIM1/3) in order to reset, start, stop, or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 35.3.6: External trigger input* for TIM1
- *Table 348: Interconnect to the tim_etr input multiplexer* for the internal ADC1/2 sources connected to TIM1 (tim_etr) input multiplexer
- *Table 372: Interconnect to the tim_etr input multiplexer* for the internal ADC1/2 sources connected to TIM3 (tim_etr) input multiplexer
- *Section 22.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for the ADCn/ADC_AWDy_OUT signal output generation

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

Active power mode

ADC1/2 are active in Run and Sleep modes.

14.3.4 Triggers to DAC

From timer (TIM1/2/4/6/7/15), LP timers (LPTIM1/3) and EXTI to DAC (DAC1)

Purpose

General-purpose timers (TIM2/4/15), basic timers (TIM6/7), advanced control timers (TIM1), LP timers (LPTIM1/3) outputs channels (lptim1_ch1 and lptim3_ch1), and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) on the TIMx_TRGO signal and from LP timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:

- *Table 197: DAC interconnection*
- *Section 23.4.8: DAC trigger selection*

Active power mode

This interconnect is active in Run, Sleep, Stop 0, Stop 1, and Stop 2 modes.

14.3.5 Triggers to ADF1

ADF1 can be triggered by EXTI15, tim1_trgo, tim3_trgo, and tim6_trgo.

Purpose

EXTI15, tim1_trgo, tim3_trgo, and tim6_trgo can trigger ADF1.

Triggering signals

ADF trigger inputs connections are detailed in:

- *Table 224: ADF trigger connections*

Active power mode

This interconnection remains active down Stop 0, Stop 1,and Stop 2 modes for ADF1, assuming the trigger source remains active.

14.3.6 Data input to ADF1

ADF1 peripheral can directly use data input from ADC1/2.

Purpose

ADC1/2 data stream can be routed to ADF1 for direct post-processing.

Triggering signals

ADF trigger inputs connections are detailed in:

- *Table 224: ADF trigger connections*

Active power mode

This interconnection is active in Run and Sleep modes.

14.3.7 Clock sources to timers

From HSE, LSE, LSI, MSIK, HSI and MCO to timers (TIM1/2/3/4/15/16/17) and LP timers (LPTIM1/2/3)

Purpose

A timer input or timer counter can receive different clock sources and can be used to calibrate internal oscillator on a reference clock for example.

External clocks (HSE, LSE), internal clocks (LSI, MSI, HSI), and microcontroller output clock (MCO) can be used as input to timers:

- MSIK/HSI are assigned to advanced-control timer TIM1 as external trigger signals inputs (tim_etr3/ tim_etr4). MSIK/HSI can be selected as counter clock provided by an external clock source in mode2: external trigger input (tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 35.3.7: Clock selection* for TIM1
  - *Table 348: Interconnect to the tim_etr input multiplexer* for TIM1
- MSIK, HSI, and LSE are assigned to general purpose timers TIM2/3/4 as external inputs signals. MSIK/HSI/LSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 36.4.5: Clock selection* for TIM2/3/4
  - External clock mode2: *Table 348: Interconnect to the tim_etr input multiplexer*, tim_etr3 (MSIK), tim_etr4 (HSI) for TIM2/3/4, tim_etr5 (MSIS) for TIM4 and tim_etr11 (LSE) for TIM2.
- LSE, LSI, MSI, and HSI are assigned to general purpose timers TIM15/16/17 as external inputs signals. LSE/LSI/MSI/HSI can be selected as counter clock provided by an external clock source in mode1 (tim_ti1 or tim_ti2 signals). Inputs assignment and clock selection description are detailed in:
  - *Table 38.4.6: Clock selection* for TIM15/16/17. External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
  - *Table 391: Interconnect to the tim_ti1 input multiplexer*, tim_ti1_in1 (LSE-TIM15), tim_ti1_in5(LSE-TIM16/17), tim_ti1_in6 (LSI- TIM16/17), tim_ti1_in7/_8 (MSI-TIM16/17), and tim_ti1_in9 (HSI-TIM16/17)
- Microcontroller output clock (MCO) is connected as external input to general-purpose timers TIM16/17. This allows the calibration of the HSI16/MSI system clocks (with TIM15/16 and LSE) or LSI (with TIM16 and HSE). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator

frequency. When the low-speed external (LSE) oscillator is used, no additional hardware connections are required. This feature is given in:

  – *Section : Internal/external clock measurement with TIM15/TIM16/TIM17*
  – *Table 343: Interconnect to the tim_ti1 input multiplexer* for TIM15/TIM16/TIM17

- LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 416: LPTIM1 input capture 2 connections*.
- HSI/256, MSI/1024, and MSI/4 can be selected as input capture 2 to LPTIM2 as described in*Table 416: LPTIM1 input capture 2 connections*.

Triggering signals

lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in).

Active power mode

This feature is available under Run and Sleep modes.

14.3.8 Triggers to low-power timers

From comparators (COMP1/2), EXTI, TAMP and RTC alarm to LP timers (LPTIM1/2/3/4)

Purpose

LPTIM1/2/3/4 counters can be started either by software or after the detection of an active edge on one of the eight trigger inputs (see *Section 39.4.7: Trigger multiplexer*).

GPIO can also be selected as LPTIM input capture selection or LPTIM input selection, according to the LPTIM configuration register 2 (LPTIM_CFGR2).

Triggering signals

This trigger feature is described in *Section 39.4.7: Trigger multiplexer* and the following sections. The input selection is described in *Section 39.4.3: LPTIM input and trigger mapping*.

Active power mode

This interconnection remains active down to Stop 2 mode.

14.3.9 Blanking sources to comparators

From timers (TIM1/2/3/4/15/16/17) to comparators (COMP1/2)

Purpose

Advanced-control timers (TIM1) and general-purpose timers (TIM2/3/4/15/16/17) can be used as blanking window input to COMP1/2.

The blanking function is described in *Section 25.3.6: Comparator output-blanking function*.

The blanking sources are given in:

- *Table 212: COMP1 output-blanking PWM assignment*, BLANKSEL[4:0]

- *Table 213: COMP2 output-blanking PWM assignment*, BLANKSEL[4:0]

Triggering signals

Timer output signal TIMx_Ocx are the inputs to blanking source of COMP1/2.

Active power mode

This feature is available under Run and Sleep modes.

14.3.10 RTC wake-up as inputs to timers

From RTC to timers (TIM16/17)

Purpose

RTC wake-up interrupt can be used as input to general-purpose timers (TIM16/17) channel 1.

Triggering signals

RTC wake-up signal is connected to tim_ti1_in4 signal as described in *Table 391: Interconnect to the tim_ti1 input multiplexer* for TIM16/17.

Active power mode

This interconnection is active down to Stop 3 mode. Timers are not active but the count is performed at wake-up.

14.3.11 USB SOF as trigger to timers

From USB SOF to TIM2

Purpose

USB SOF (start-of-frame) can generate a trigger to the general-purpose timer TIM2. The USB connection to TIM2 is described in *Table 372: Interconnect to the tim_etr input multiplexer*.

Triggering signals

The tim_etr13 internal signal is generated by USB SOF.

Active power mode

This interconnection is active in Run and Sleep modes.

14.3.12 Comparators as inputs, trigger or break signals to timers

From comparators to timers (TIM1/2/3/4/15/16/17)

Purpose

The comparators (COMP1/2) output values can be connected to timers (TIM1/2/3/4/15/16/17) input captures, TIMx_ETR or timer break signals. The connection to ETR is described in *Section 35.3.6: External trigger input*.

Comparators (COMP1/2) output values can also generate break input signals for timers (such as TIM1). The sources for break (tim_brk) channel are one of the following:

- external: connected to one of the TIMx_BKIN pin (as per selection done in the AFIO controller) with polarity selection and optional digital filtering
- internal: coming from comparators, tim_brk_cmpx input (refer to *Table 342: TIM internal input/output signals* for product specific implementation).

Triggering signals

The tim_etr and tim_brk signals connected TIM1 (coming from COMP1/2) are given in:

- tim_etr (*Table 348: Interconnect to the tim_etr input multiplexer*): external trigger internal input bus

These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control.

- tim_brk (*Table 349: Timer break interconnect* and *Table 350: Timer break2 interconnect*)
- *Section 35.3.6: External trigger input*
- *Section 35.3.18: Using the break function*

For TIM2/3/4, the sources connected to the tim_ti[1:4] input multiplexers coming from comparators and some other peripherals, are given in:

- *Table 367: Interconnect to the tim_ti1 input multiplexer*
- *Table 368: Interconnect to the tim_ti2 input multiplexer*
- *Table 370: Interconnect to the tim_ti4 input multiplexer*
- *Table 372: Interconnect to the tim_etr input multiplexer*
- *Table 373: Interconnect to the tim_ocref_clr input multiplexer*
- *Section 36.4.22: Timers and external trigger synchronization*
- TIMx_TISEL

For TIM15/16/17, the sources connected to timers coming from comparators and other peripherals are given in:

- *Table 392: Interconnect to the tim_ti2 input multiplexer*
- *Table 394: Timer break interconnect*
- *Table 396: Interconnect to the ocref_clr input multiplexer*
- *Section 38.4.15: Using the break function*
- *Section 38.4.23: External trigger synchronization (TIM15 only)*

Active power mode

Run, Sleep, and wake-up capability in Stop 0, Stop 1, and Stop 2 modes for trigger sources. Input and break remain active in same low-power modes as timers activity, on Run and Sleep modes.

14.3.13 System errors as break signals to timers

From system errors to timers (TIM1/15/16/17)

Purpose

CSS, CPU lockup, SRAM2 parity errors, ECC double errors, FLASH ECC double-error detection, and PVD can generate system errors in the form of timer break toward timers (TIM1/15/16/17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

Triggering signals

The possible sources of break are described in:

- *Section 35.3.18: Using the break function* for TIM1
- *Section 38.4.15: Using the break function* for TIM15/16/17
- *Table 351: System break interconnect* for TIM1
- *Table 395: System break interconnect* for TIM15/16/17

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 360: Effect of low-power modes on TIM1*
- *Table 379: Effect of low-power modes on TIM2/TIM3/TIM4*
- *Table 400: Effect of low-power modes on TIM15/TIM16/TIM17*

14.3.14 Timers generating IRTIM signal

From timers (TIM16/17) to IRTIM

The general-purpose timer (TIM16/TIM17) output channels TIMx_OC1 and USART1/UART4 TX output, are used to generate the waveform of the infrared signal output.

This functionality is described in *Section 40: Infrared interface (IRTIM)*.

14.3.15 Triggers for communication peripherals

From LP timers (LPTIM1/2/3), comparators (COMP1/2), GPDMA1 transfer complete, EXTI, RTC alarm and RTC wake-up to USART1/3, UART4/UART5, LPUART1, I2C1/2, I3C1/2, and SPI1/2

Purpose

LP timer (LPTIM1/2/3) output channels (lptim1_ch1, lptim1_ch2 and lptim3_ch1), timers (tim15_trgo_cktim, tim16_oc1), comparator (COMP1/2) output channels (comp1_out and comp2_out),

EXTI, RTC alarm and RTC wakeup, can be used as trigger to start a communication on the selected USART, UART, LPUART, I2C, I3C, or SPI peripherals.

A GPDMA1 transfer complete can trigger both the GPDMA1 regular or linked-list new transfers and communication on selected communication peripheral.

Triggering signals

The outputs from triggers are directly connected to peripheral trigger inputs.

The selection of input triggers is detailed in:

- *Table 495: USART interconnection (USART1/3 and UART4/5)*
- *Table 507: LPUART interconnections (LPUART1)*
- *Table 457: Trigger interconnections* for I2C peripherals
- *Table 477: I3C triggers, when I3C as controller* for I3C peripherals

The outputs (from timer) are directly connected to SPI1/2 inputs on signals spi_itrx (x = 6, 7). The selection of input triggers on SPI is provided in:

- *Section : Description of SPI interconnections*

Active power mode

These interconnections remain active in Run, Sleep, and Stop modes if both source and communication line are autonomous under the mode. Refer to:

- *Section 47.6: USART in low-power modes*
- *Section 45.5: I2C in low-power modes*
- *Section 49.6: SPI in low-power modes*
- *Section 46.14: I3C in low-power modes*

14.3.16 Triggers to GPDMA

From EXTI, RTC (alarm/wakeup), TAMP (TAMP1/2/3), timers (TIM2/15), LP timers (LPTIM1/3/4), comparators (COMP1/2), GPDMA1 transfer complete (gpdma1_chx_tc), ADC1/2 and analog watchdog to GPDMA1

Purpose

A GPDMA trigger can be assigned to a GPDMA channel x. A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- *Section 15.3.6: GPDMA triggers*
- *Section 15.4.12: GPDMA triggered transfer*
- GPDMA_CxTR2 for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0].
  - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

GPDMA trigger mapping is specified in *Table 122: Programmed GPDMA1 trigger*, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

Assuming sources are active down to Stop modes, this interconnection remains functional in Stop 0 and Stop 1 modes for GPDMA.

Refer to:

- *Section 15.6: GPDMA in low-power modes*

14.3.17 Internal analog signals to analog peripherals

From internal analog source to ADC (ADC1/2), comparators (COMP1/2) and OPAMP (OPAMP1/2)

Purpose

The internal reference voltage (V_{REFINT}), the internal temperature sensor (V_{TS}), and V_{BAT} monitoring channel are connected to ADC (ADC1/2) input channels. In addition, the internal digital core voltage (V_{CORE}) is connected /to ADC1/2 input channels.

DAC channels (DAC1_OUT1/DAC2_OUT2) and V_{REFINT} are connected to comparators (COMP1/2).

OPAMP1/2 outputs can be connected to ADC1/2 input channels through the GPIO. DAC1_OUT1 can be connected to OPAMP1_VINP. DAC1_OUT2 can be connected to OPAMP2_VINP.

Refer to *Section 23.4.2: DAC pins and internal signals* for:

- dac_out1 analog output DAC channel1, output, for on-chip peripherals
- dac_out2 analog output DAC channel2, output, for on-chip peripherals

This is according to:

- *Section 22.2: ADC main features*, Figure 119. ADC1 connectivity, Figure 120. ADC2 connectivity
- *Section 26.3.3: Signal routing*
- *Section 25.3.2: COMP pins and internal signals*

Active power mode

These interconnections remain in Stop modes if the selected peripheral is kept active. Refer to:

- *Section 22.5: ADC in low-power modes*
- *Section 26.4: OPAMP low-power modes*
- *Section 25.4: COMP in low-power modes*

14.3.18 Clock source for the DAC sample and hold mode

From LSI/LSE to DAC1/2

Purpose

DAC1 can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

Section 23.4.2: DAC pins and internal signals:

dac_hold_ck (input) - DAC low-power clock used in sample and hold mode

Active power mode

This feature remains available down to Stop 2 mode.

14.3.19 Internal tamper sources

From internal peripherals, clocks to tamper.

Purpose

In order to detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system of such undesired event. Different actions can be taken in consequences.

List of tamper sources can be found in *Table 446: TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

14.3.20 Output from tamper to RTC

From TAMP to RTC

Purpose

The RTC can timestamp a tamper event in order to retrieve history in time of such detection. The RTC can also control GPIOs and set a signal based on tamp or alarm status outside the MCU.

Refer to *Section 43.3.3: GPIOs controlled by the RTC* for more details.

Active power mode

This interconnection remains active in all power modes.

14.3.21 Encryption keys to AES/SAES

From TAMP backup registers, system flash memory to and in between SAES and AES

Purpose

The encryption mechanism requires an hardware key that must be stored in a protected non-volatile memory. Different approaches are implemented in order to load them in a non-readable way. Tamper backup registers or system flash memory can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to *Section 32.4.14: SAES operation with wrapped keys* for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to *Section 32.4.15: SAES operation with shared keys* for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.

14.3.22 I3C IBI acknowledge to timers from I3C1/2 to TIM15/16/17 and LPTIM1/2/3

Purpose

The I3C peripherals generate an IBI acknowledge event when peripheral acts as controller, for timing control with asynchronous mode 0. This enables to timestamp the moment at which the target(s) acquired sampled data and to time-correlate them.

Refer to *Section 46.9.13: I3C IBI transfer, as controller/target* for more details.

Active power mode

Timers are functional in Run and Sleep modes. I3C is functional in Run, Sleep, Stop 0, and Stop 1 modes, and so are functional LP timers.

15 General purpose direct memory access controller (GPDMA)

15.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

15.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 12 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 15.3.1*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 15.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

15.3 GPDMA implementation

15.3.1 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 116. GPDMA1 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 7	2	0	Channel x (x = 0 to 7) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 8 to 9	4	0	Channel x (x = 8 to 9) is implemented with: – a FIFO of 32 bytes, 8 words – fixed/contiguously incremented addressing
x = 10 to 11	4	1	Channel x (x = 10 to 11) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be also used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

15.3.2 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 117. GPDMA1 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1 in Sleep mode

15.3.3 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[6:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 118. Programmed GPDMA1 request

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
0	adc1_dma
1	adc2_dma
2	dac1_dma
3	dac2_dma
4	tim6_up_dma
5	tim7_up_dma
6	spi1_rx_dma

Table 118. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
7	spi1_tx_dma
8	spi2_rx_dma
9	spi2_tx_dma
10	spi3_rx_dma
11	spi3_tx_dma
12	i2c1_rx_dma
13	i2c1_tx_dma
14	i2c1_evc_dma
15	i2c2_rx_dma
16	i2c2_tx_dma
17	i2c2_evc_dma
18	i2c3_rx_dma
19	i2c3_tx_dma
20	i2c3_evc_dma
24	usart1_rx_dma
25	usart1_tx_dma
28	usart3_rx_dma
29	usart3_tx_dma
30	uart4_rx_dma
31	uart4_tx_dma
32	uart5_rx_dma
33	uart5_tx_dma
34	lpuart1_rx_dma
35	lpuart1_tx_dma
36	sai1_a_dma
37	sai1_b_dma
38	Reserved
39	Reserved
40	octospi1_dma
41	Reserved
42	tim1_cc1_dma
43	tim1_cc2_dma
44	tim1_cc3_dma
45	tim1_cc4_dma
46	tim1_up_dma

Table 118. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
47	tim1_trig_dma
48	tim1_com_dma
49	i3c1_rx_dma
50	i3c1_tx_dma
51	i3c1_tc_dma
52	i3c1_rs_dma
53	Reserved
54	Reserved
55	Reserved
56	tim2_cc1_dma
57	tim2_cc2_dma
58	tim2_cc3_dma
59	tim2_cc4_dma
60	tim2_up
61	tim3_cc1_dma
62	tim3_cc2_dma
63	tim3_cc3_dma
64	tim3_cc4_dma
65	tim3_up
66	tim3_trig
67	tim4_cc1_dma
68	tim4_cc2_dma
69	tim4_cc3_dma
70	tim4_cc4_dma
71	tim4_up
72	i3c2_rx_dma
73	i3c2_tx_dma
74	i3c2_tc_dma
75	i3c2_rs_dma
78	tim15_cc1_dma
79	tim15_up_dma
80	tim15_trig_dma
81	tim15_com_dma
82	tim16_cc1_dma
83	tim16_up_dma

Table 118. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
84	tim17_cc1_dma
85	tim17_up_dma
86	Reserved
87	aes_in_dma
88	aes_out_dma
89	hash_in_dma
90	Reserved
98	adf1_flt0_dma
99	Reserved
100	Reserved
101	Reserved
102	Reserved
103	saes_in_dma
104	saes_out_dma
105	lptim1_ic1_dma
106	lptim1_ic2_dma
107	lptim1_ue_dma
108	lptim2_ic1_dma
109	lptim2_ic2_dma
110	lptim2_ue_dma
111	lptim3_ic1_dma
112	lptim3_ic2_dma
113	lptim3_ue_dma

15.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 119. Programmed GPDMA1 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma
lptim3_ue_dma

15.3.5 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 120. GPDMA1 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0 and x = 1

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the GPDMA has transferred the whole block (see *Section 15.4.14* for more details).

Table 121. Programmed GPDMA1 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma
i3c2_rx_dma

15.3.6 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 122.Programmed GPDMA1 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2
13	lptim2_ch1

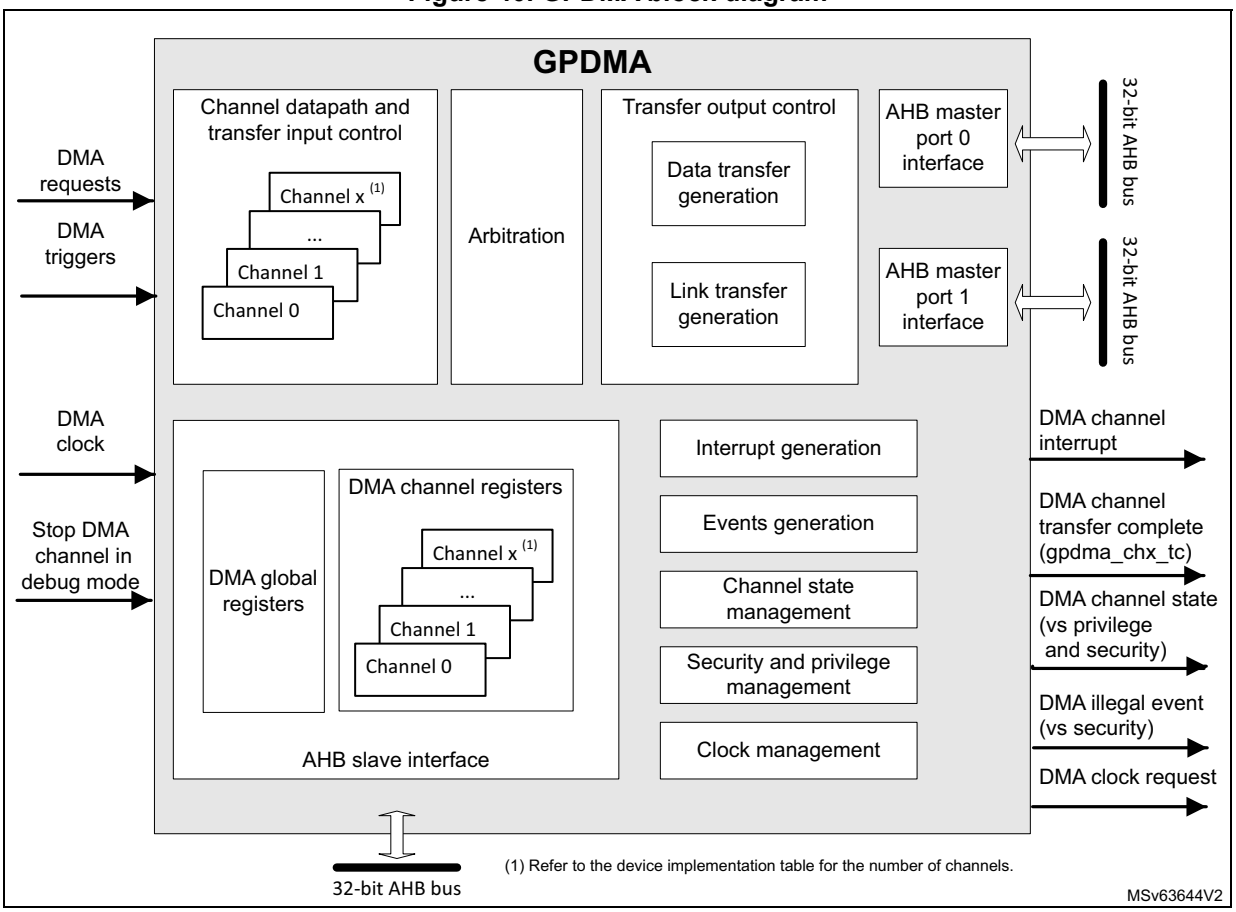
Table 122.Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
14	lptim2_ch2
15	lptim4_out
16	comp1_out
17	comp2_out
18	rtc_alra_trg
19	rtc_alrb_trg
20	rtc_wut_trg
21	Reserved
22	gpdma1_ch0_tc
23	gpdma1_ch1_tc
24	gpdma1_ch2_tc
25	gpdma1_ch3_tc
26	gpdma1_ch4_tc
27	gpdma1_ch5_tc
28	gpdma1_ch6_tc
29	gpdma1_ch7_tc
30	gpdma1_ch8_tc
31	gpdma1_ch9_tc
32	gpdma1_ch10_tc
33	gpdma1_ch11_tc
42	tim2_trgo
46	Reserved
47	Reserved
48	Reserved
49	Reserved
50	Reserved
51	Reserved
52	Reserved
53	Reserved
54	Reserved
55	Reserved
56	Reserved
57	adc2_awd1
58	adc1_awd1

15.4 GPDMA functional description

15.4.1 GPDMA block diagram

Figure 40. GPDMA block diagram



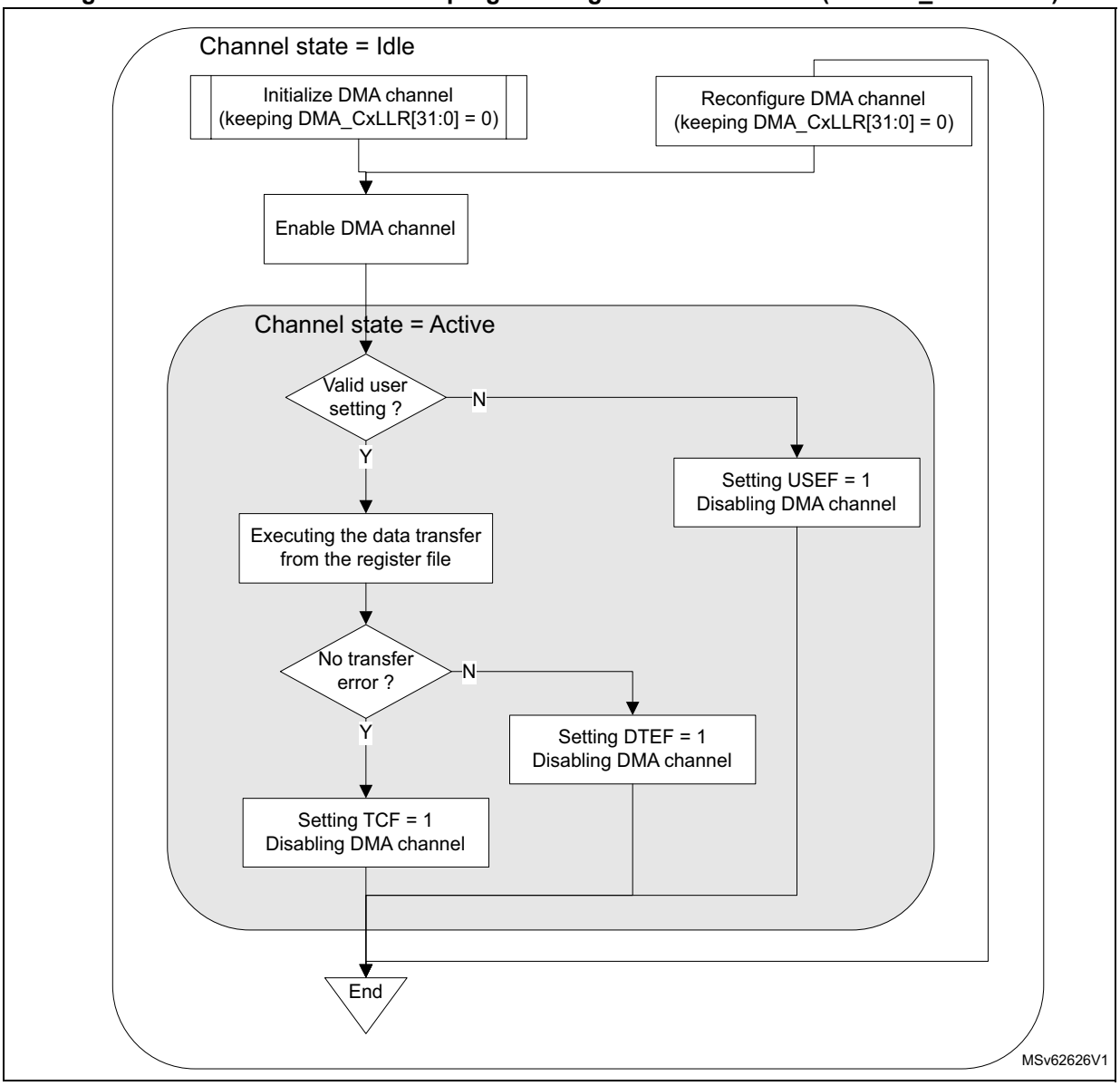
15.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 41 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 41. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



15.4.3 GPDMA channel suspend and resume

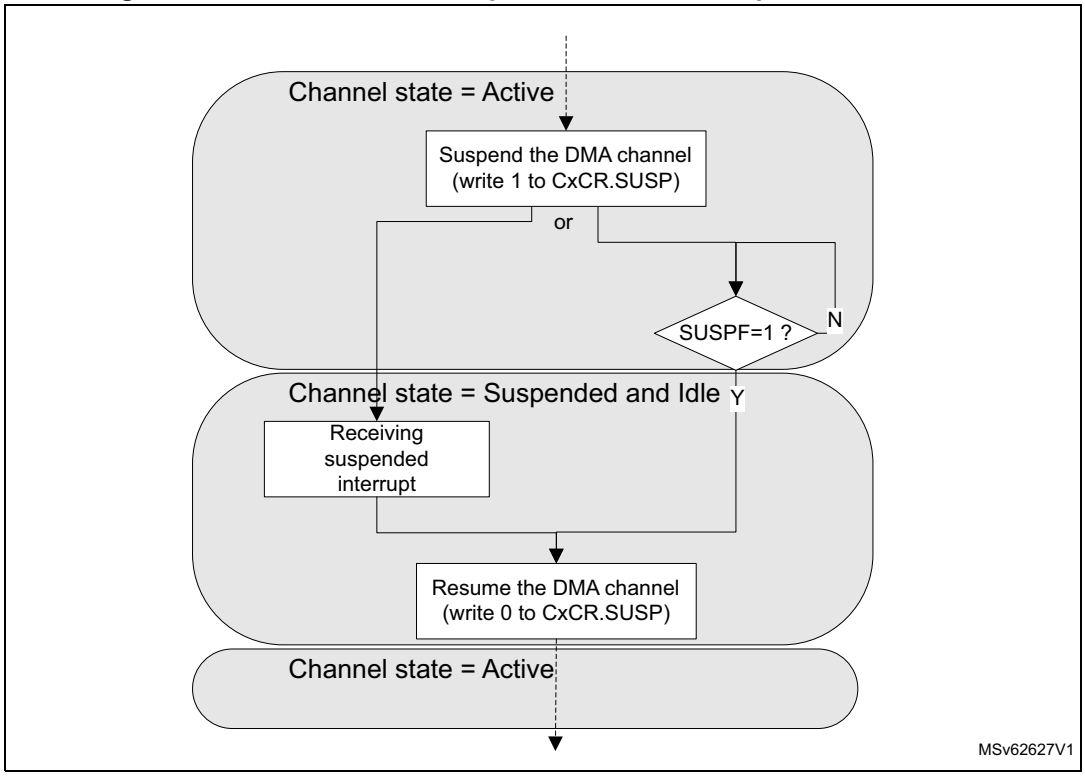
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 42. GPDMA channel suspend and resume sequence



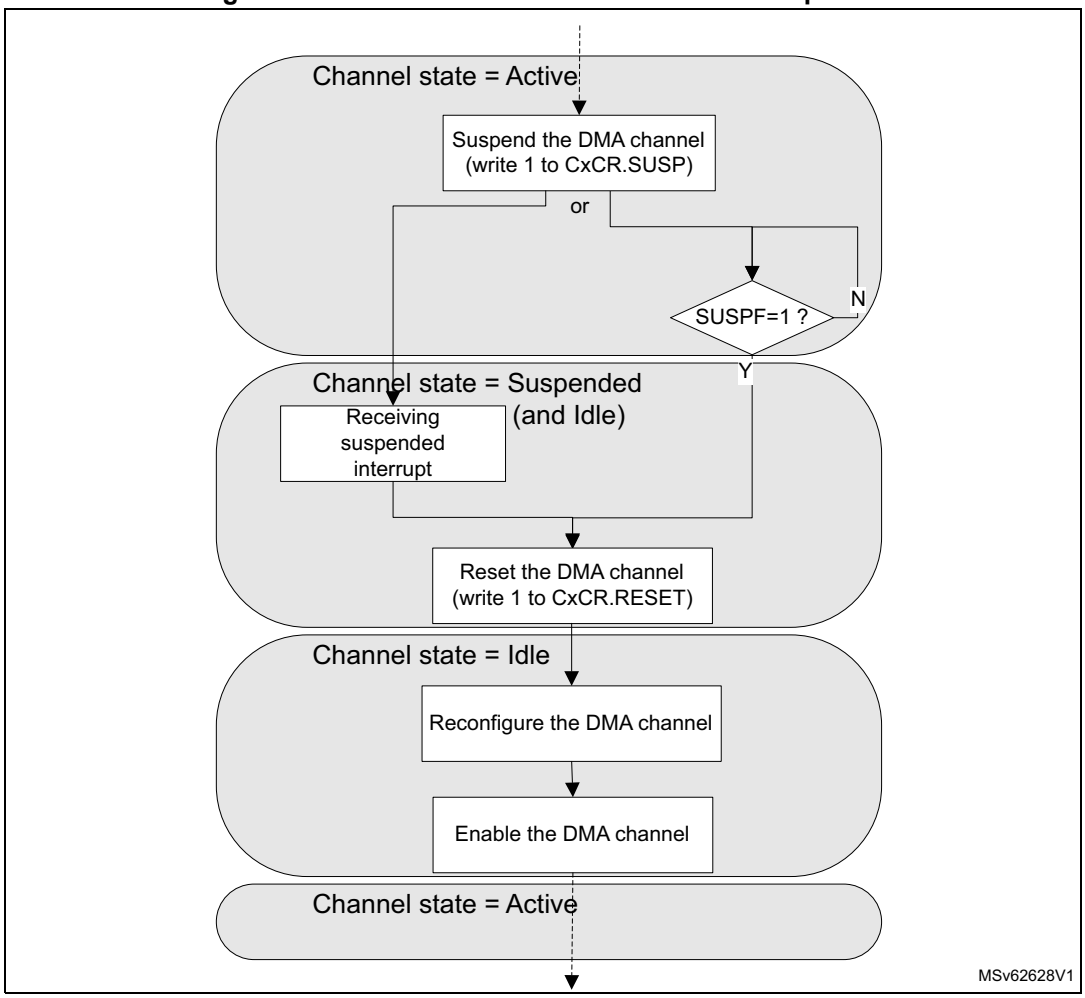
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

15.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 43. GPDMA channel abort and restart sequence



15.4.5 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the

GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 9) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 44*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 10 to 11), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see *Figure 45*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 44. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

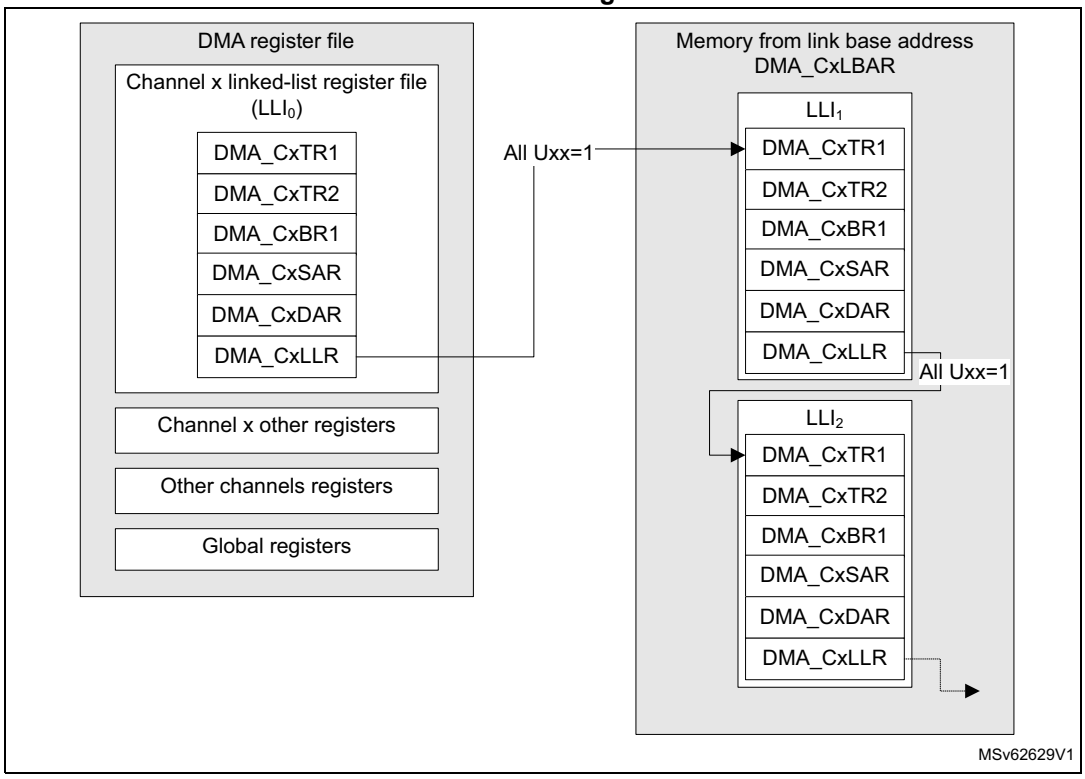
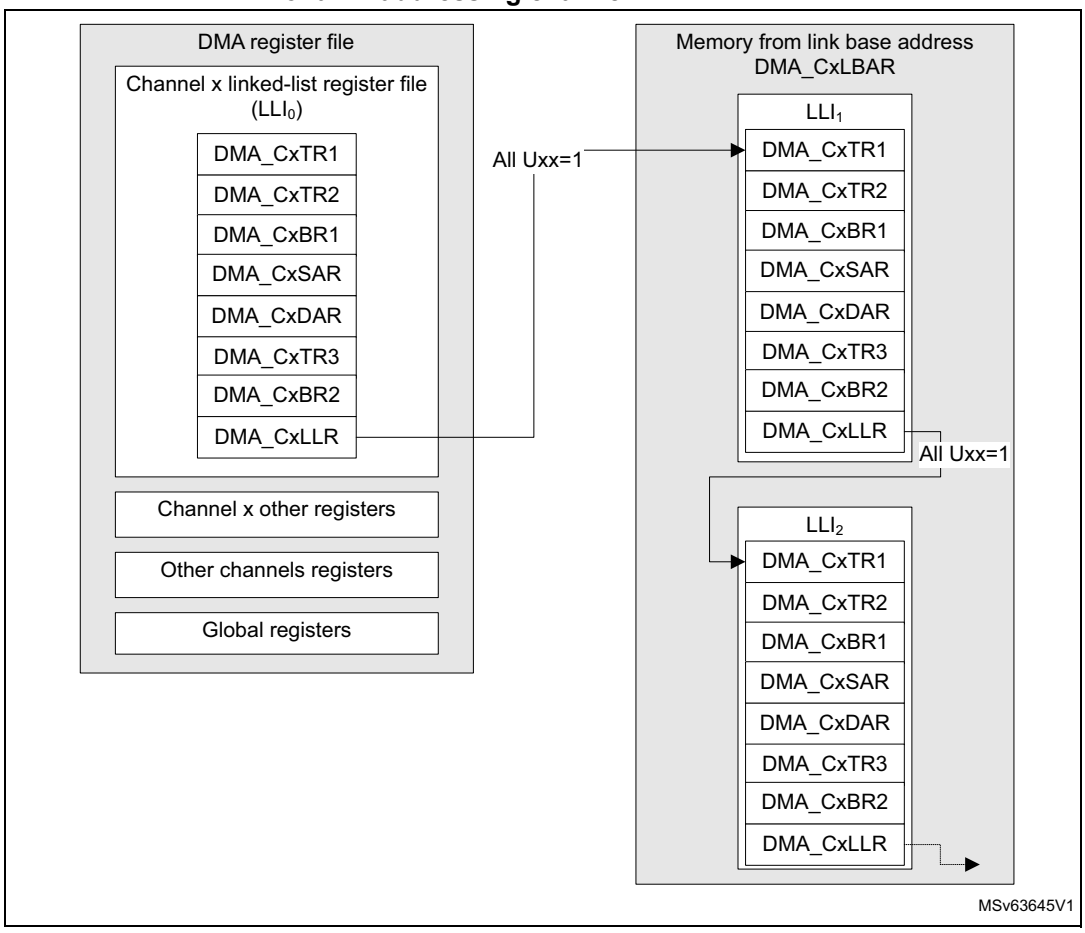


Figure 45. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 46*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 47*.

Figure 46. GPDMA dynamic linked-list data structure of a linear addressing channel x

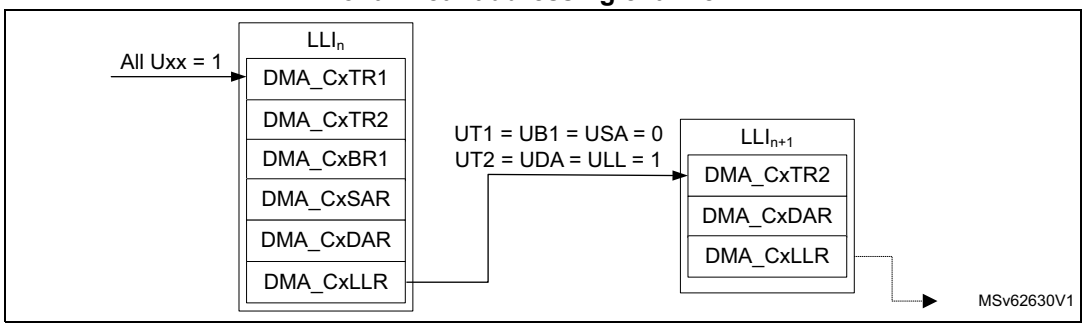
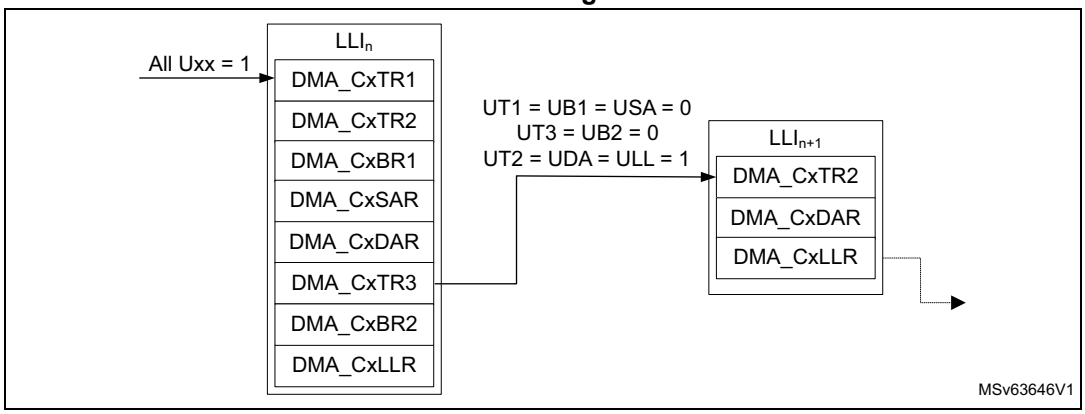


Figure 47. GPDMA dynamic linked-list data structure of a 2D addressing channel x



The user must program GPDMA_CxLLR for each LLI_n to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

15.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

15.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 15.4.14* for more details).

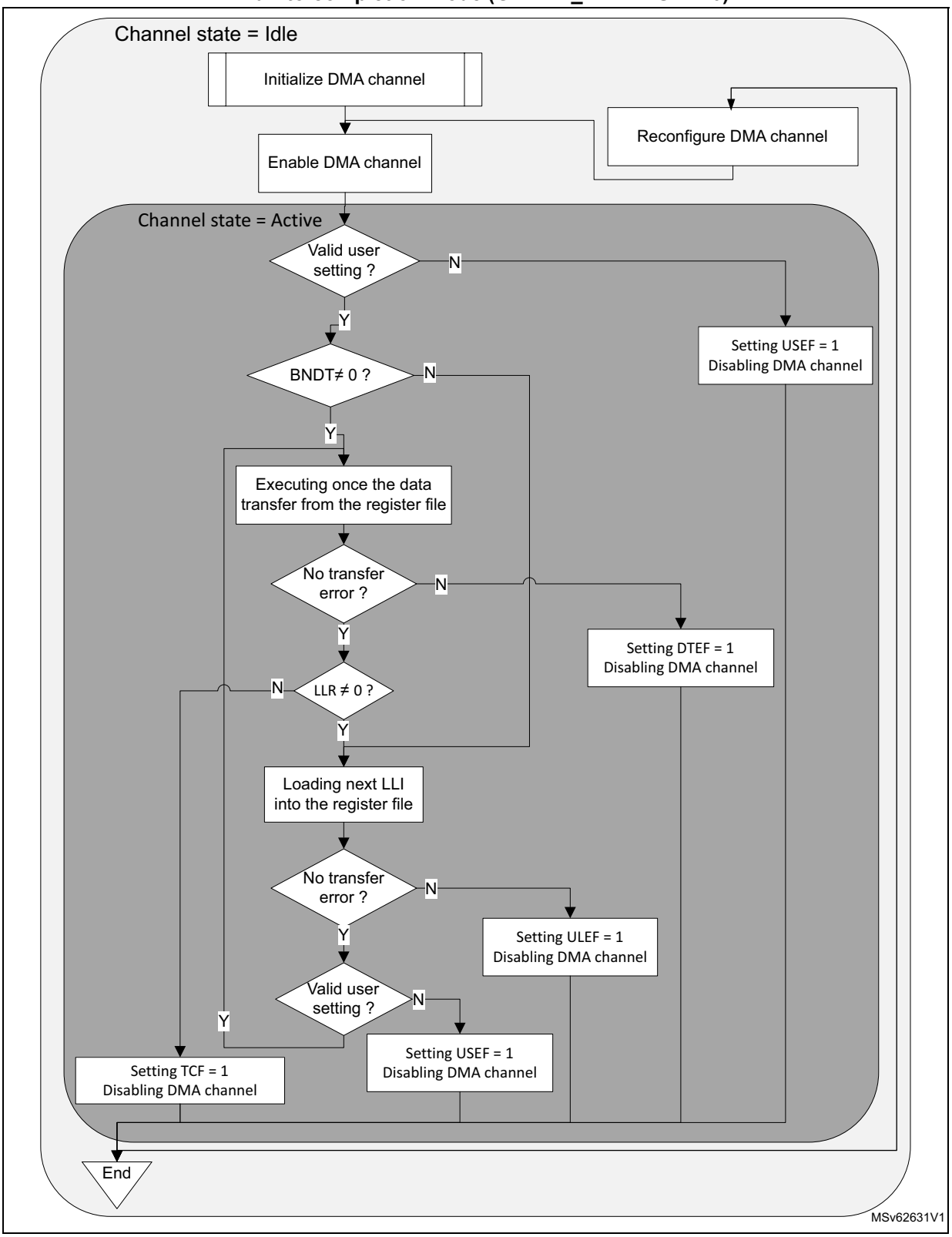
In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 48 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 48 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 48, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.5).

Figure 48. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

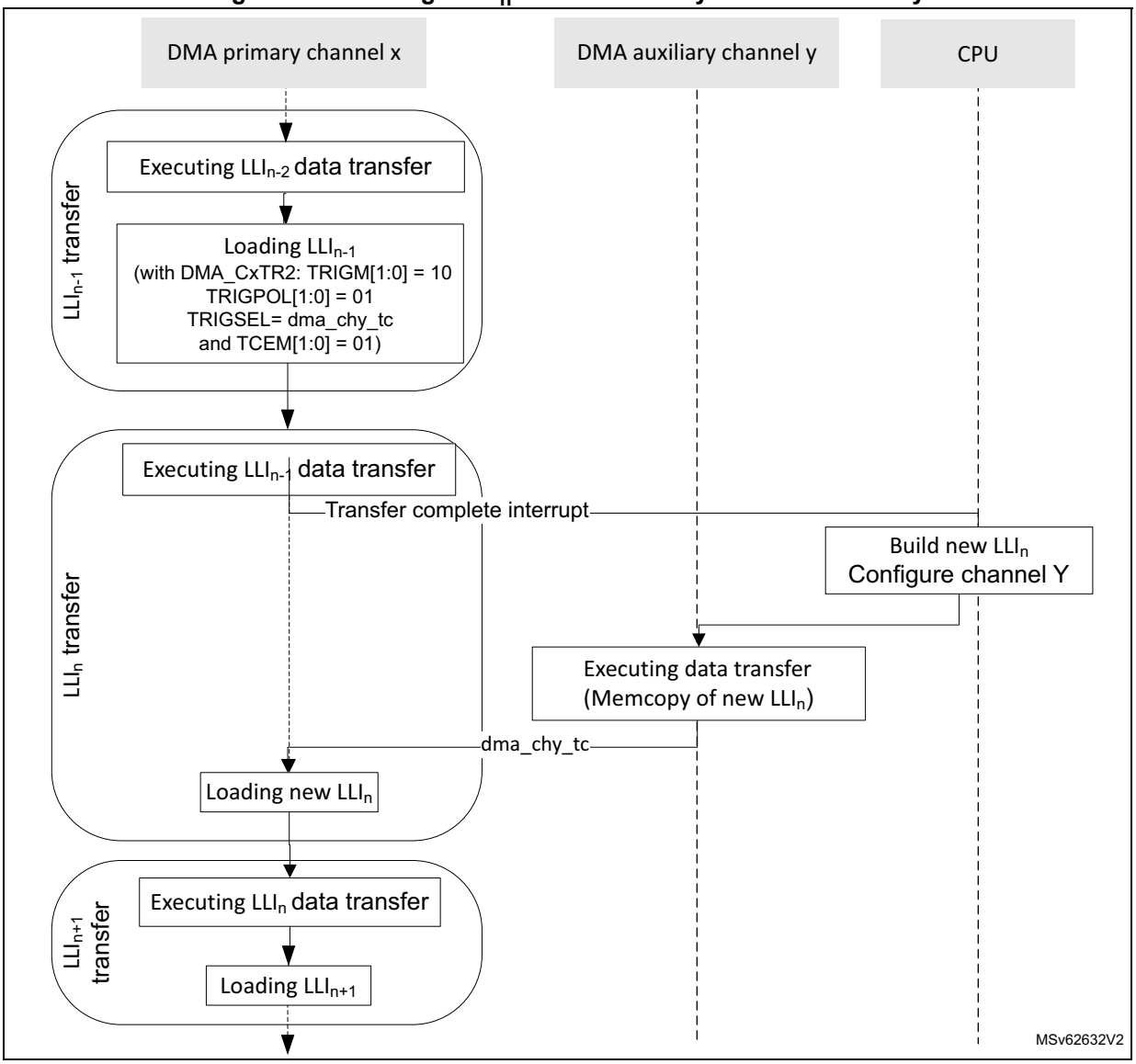
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 15.3.6* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 49 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 49. Inserting a LLI_n with an auxiliary GPDMA channel y



15.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 10 to 11), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 10 to 11) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 10 to 11) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 15.4.14* for more details).

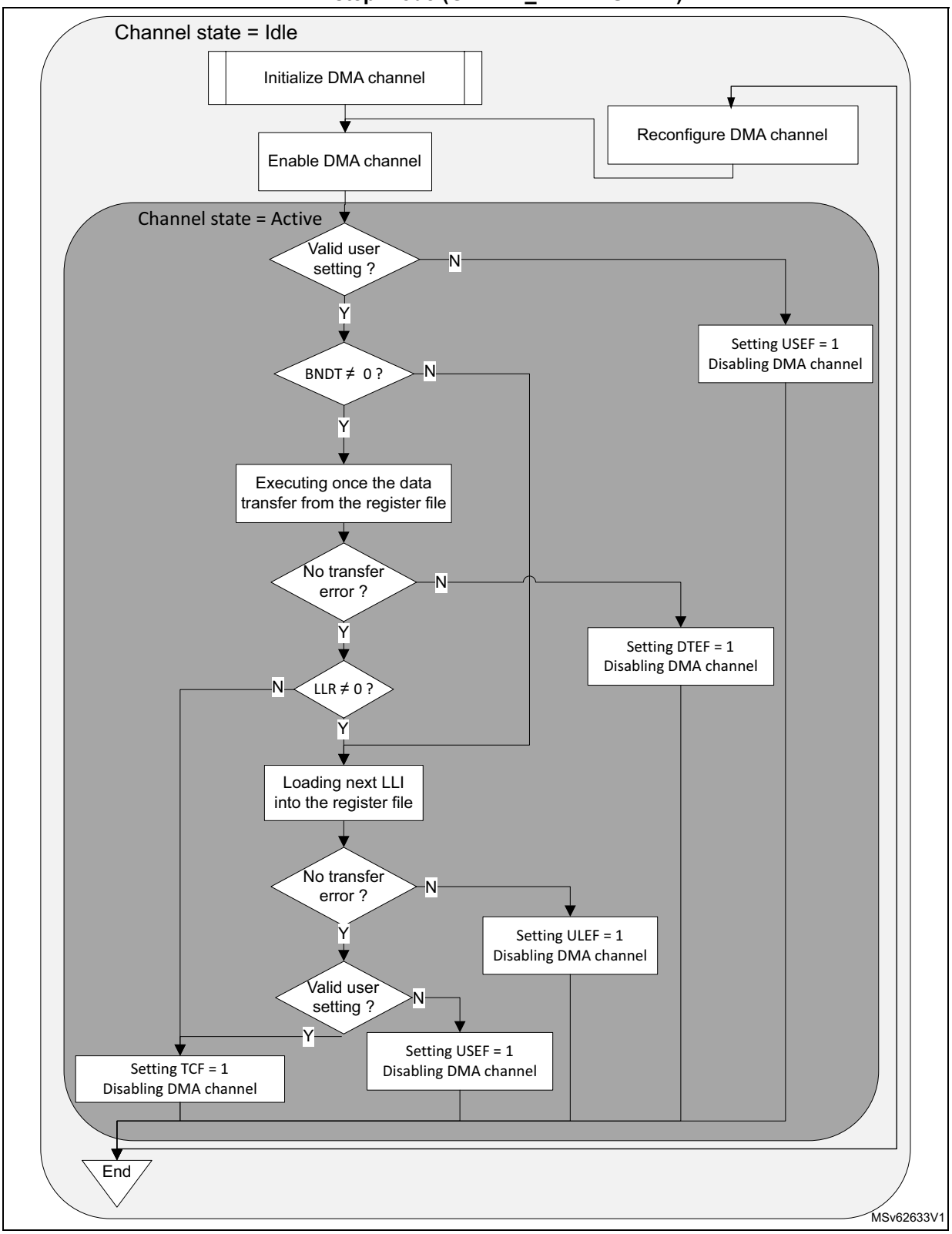
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 50 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 50 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 50, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.5).

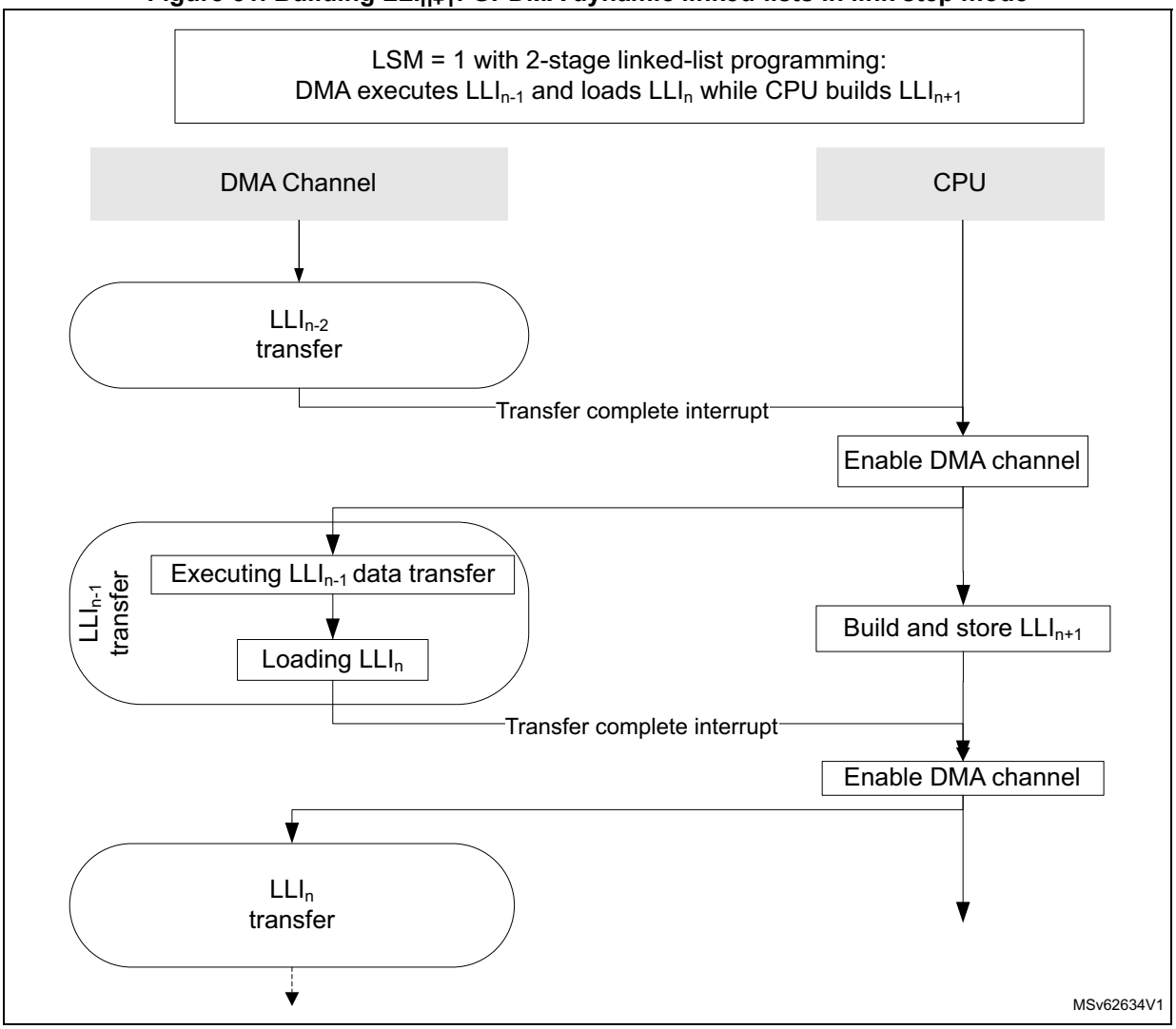
Figure 50. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 51*.

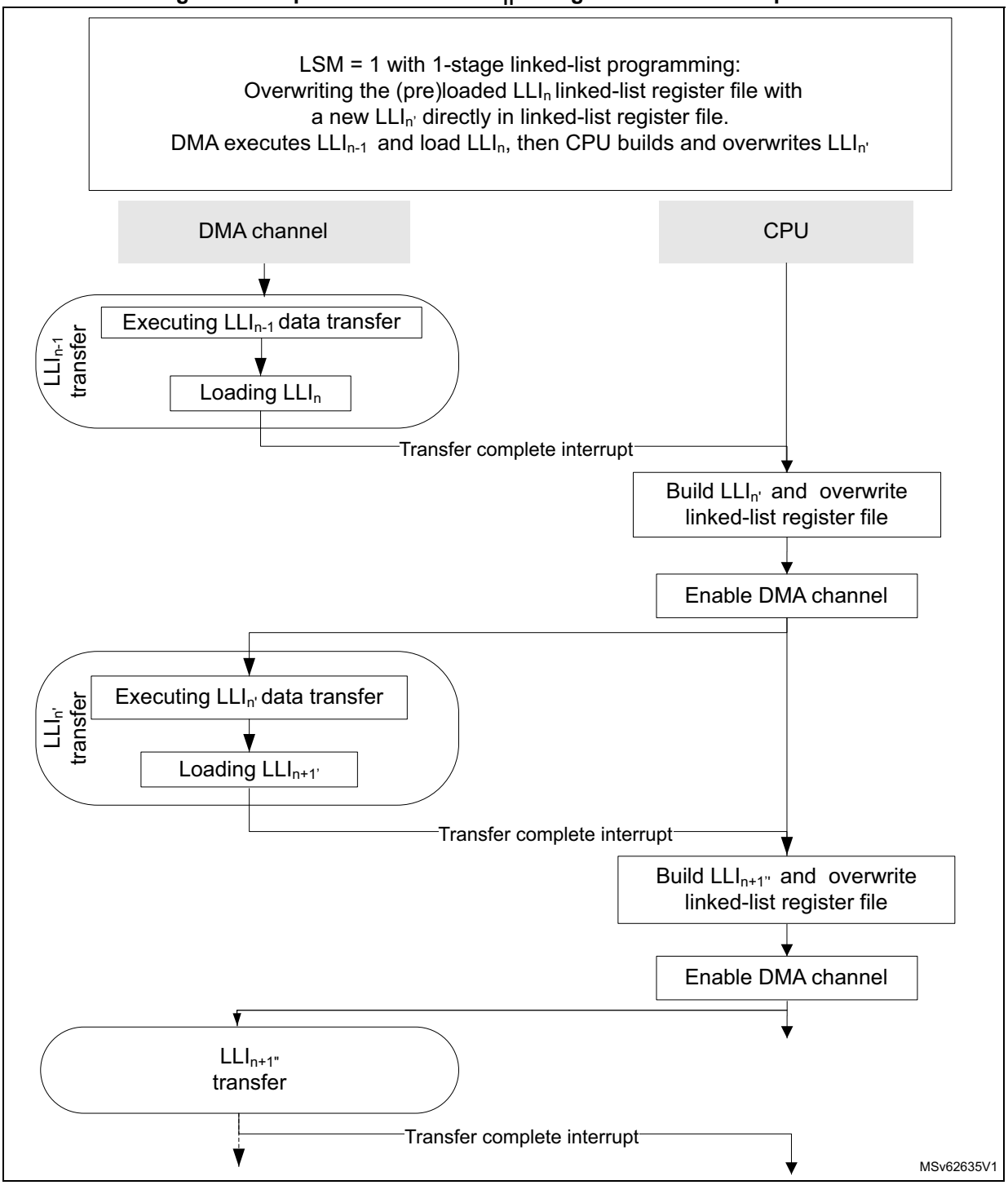
Figure 51. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new $LLI_{n'}$, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new $LLI_{n'}$, as shown in *Figure 52*.

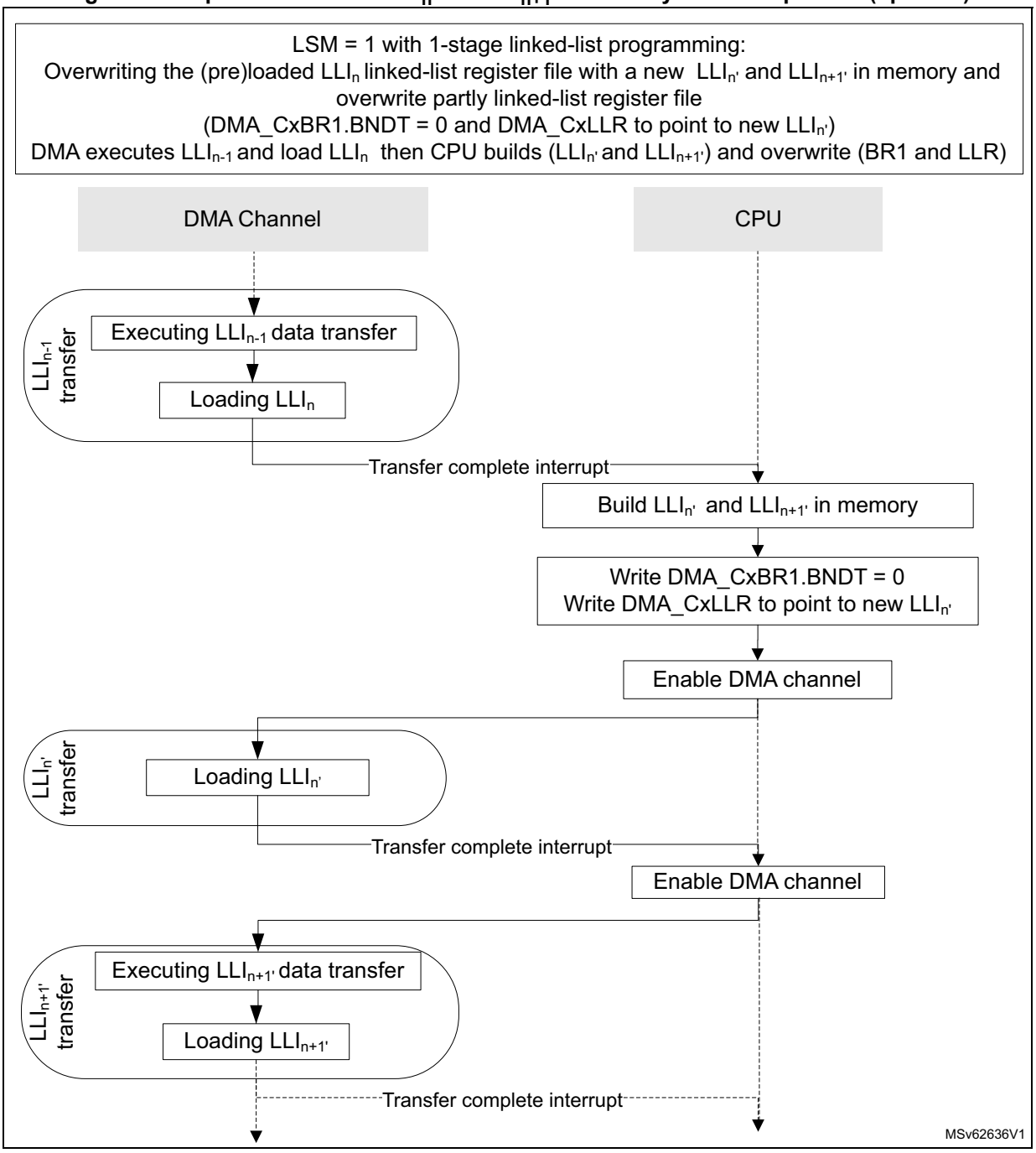
Figure 52. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 53*.

Figure 53. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



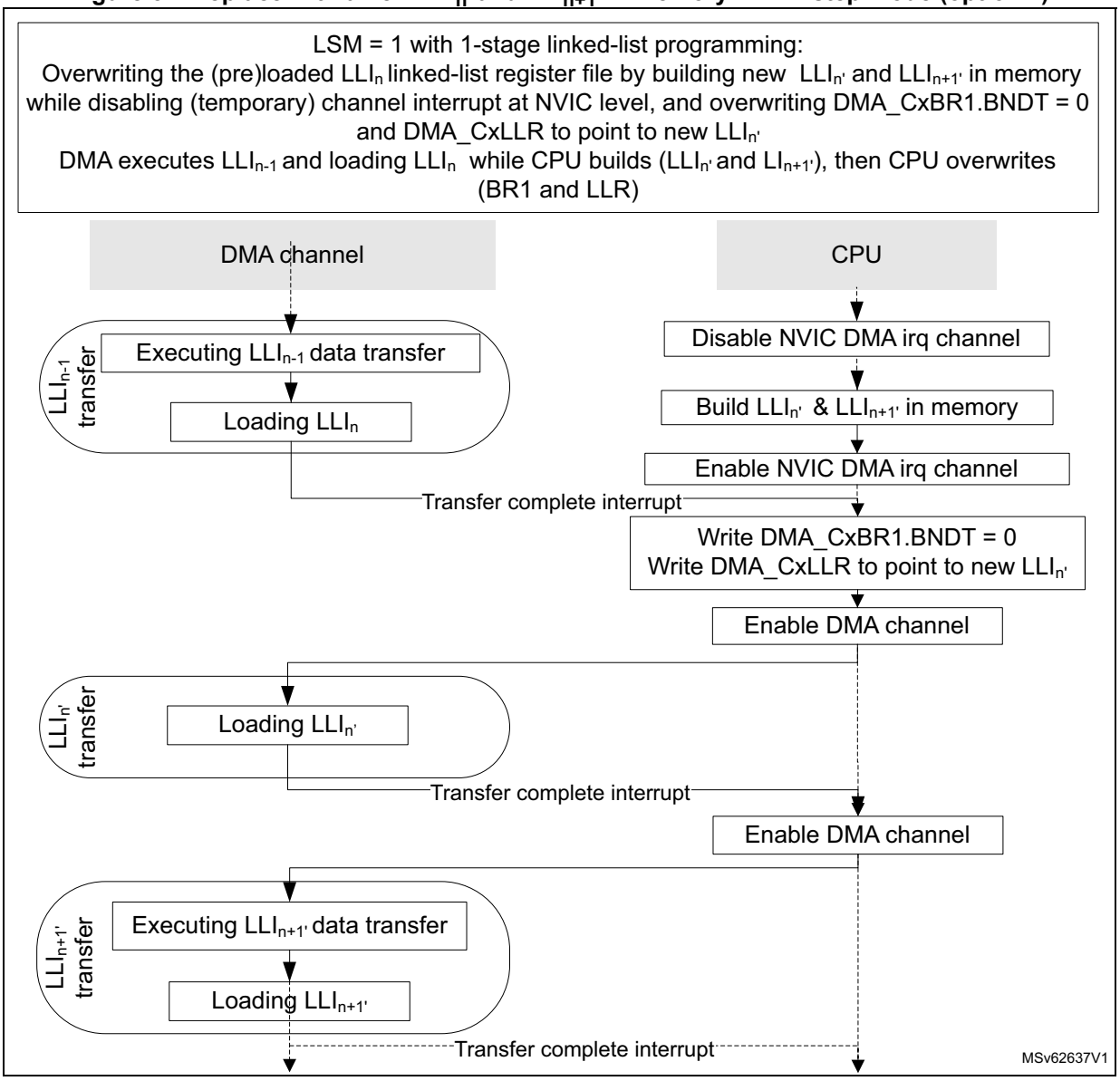
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 54*.

Figure 54. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



15.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 15.4.4*.)

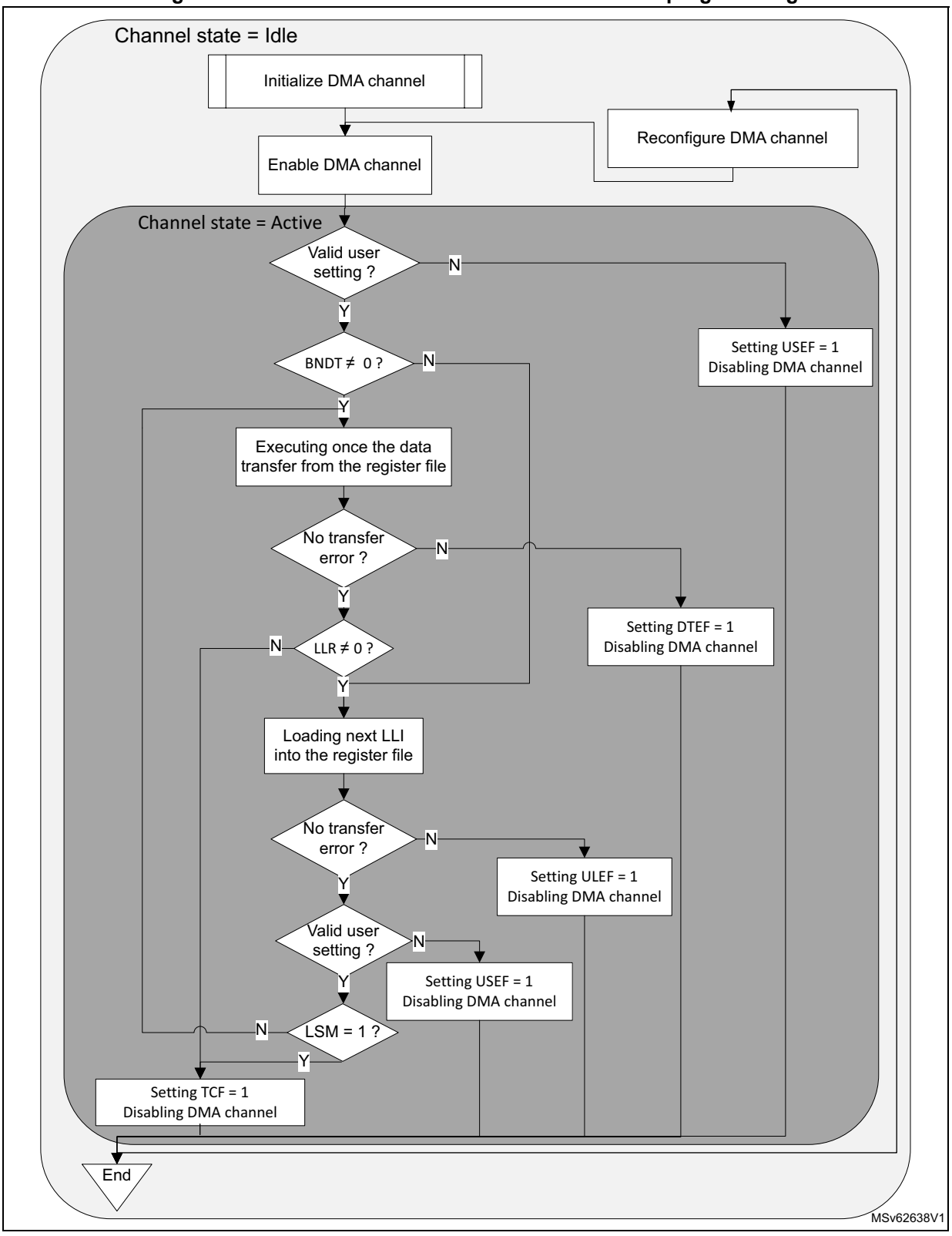
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 55 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 55 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 55, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 15.3.5).

Figure 55. GPDMA channel execution and linked-list programming



15.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 15.3.1* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 123. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

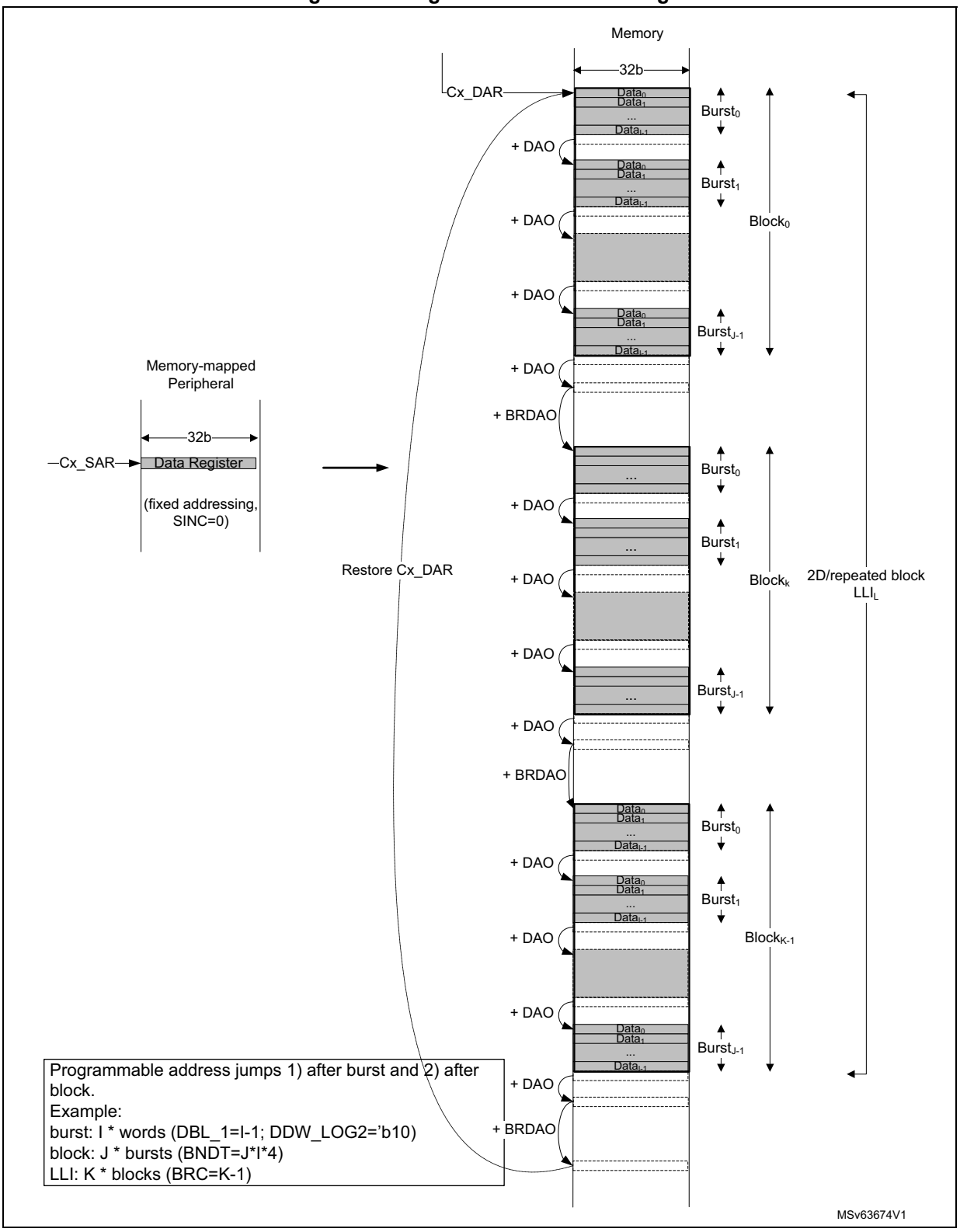
GPDMA burst with 2D addressing (channel x = 10 to 11)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 56* with a 2D destination buffer.

Figure 56. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half of the FIFO size of the channel x, in bytes (see *Section 15.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 124*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 124. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 124. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 124. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

15.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[6:0] field (see *Section 15.3.3*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[6:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 57*, is defined by:

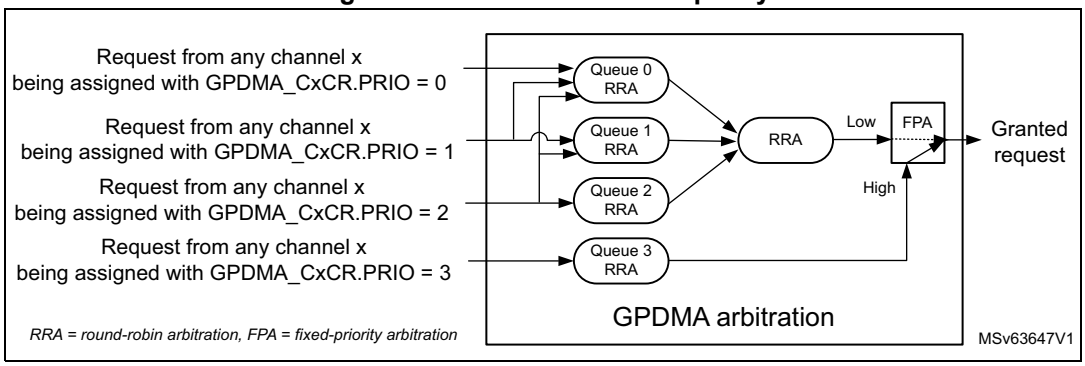
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 57. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

15.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 15.3.6* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 10 to 11), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

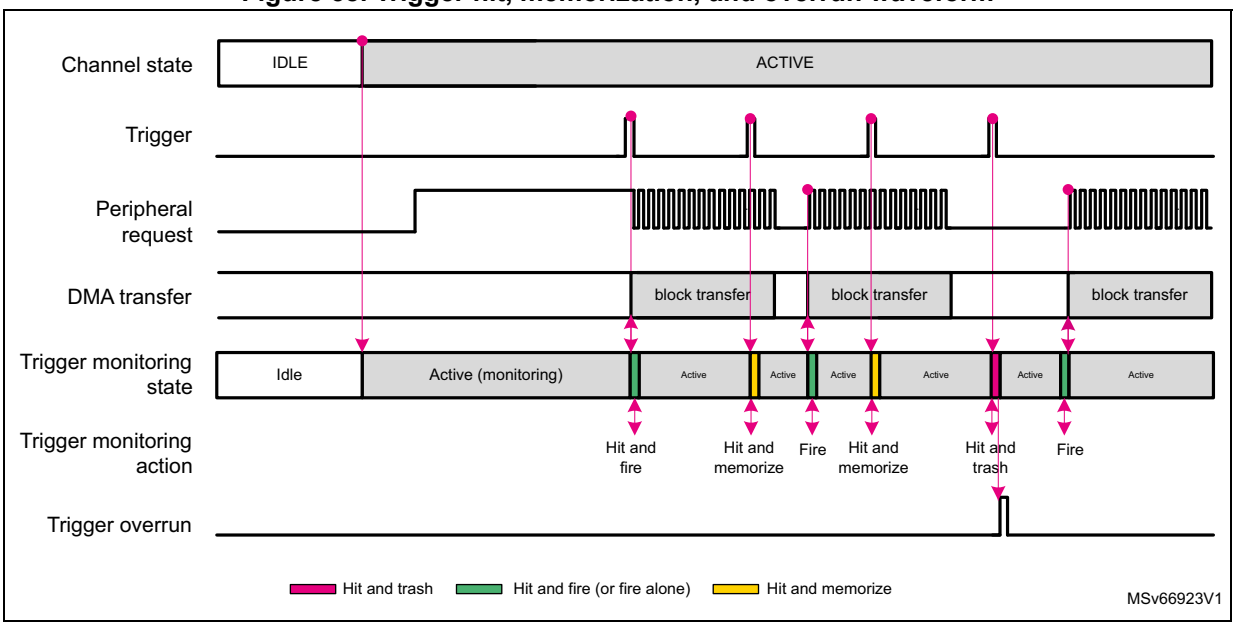
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 58 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 58. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

15.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

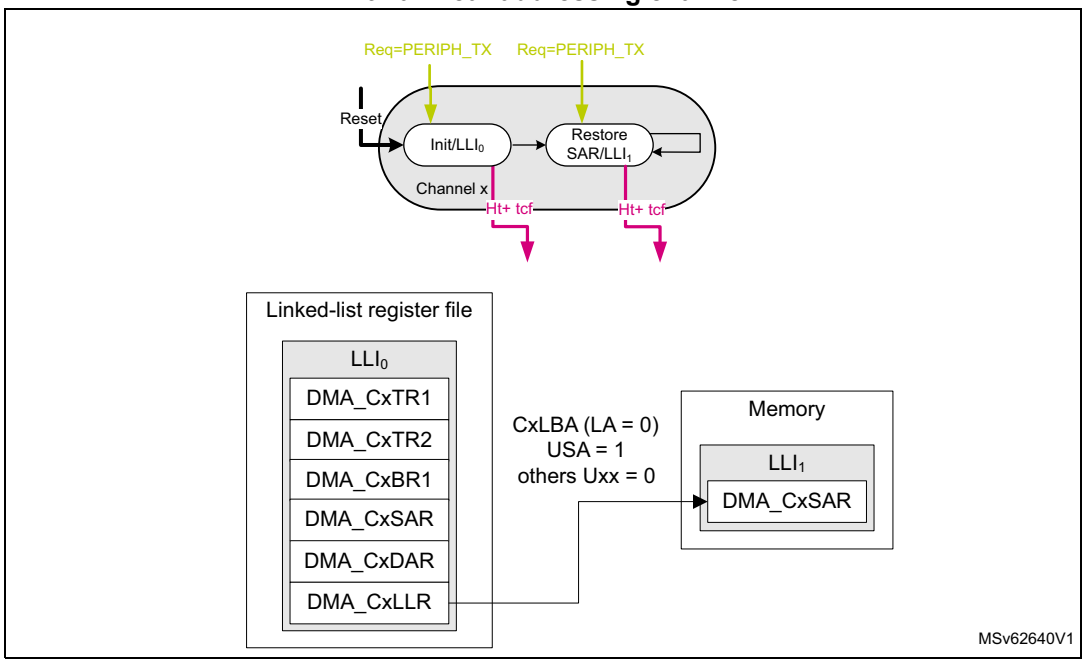
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 59 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 59. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

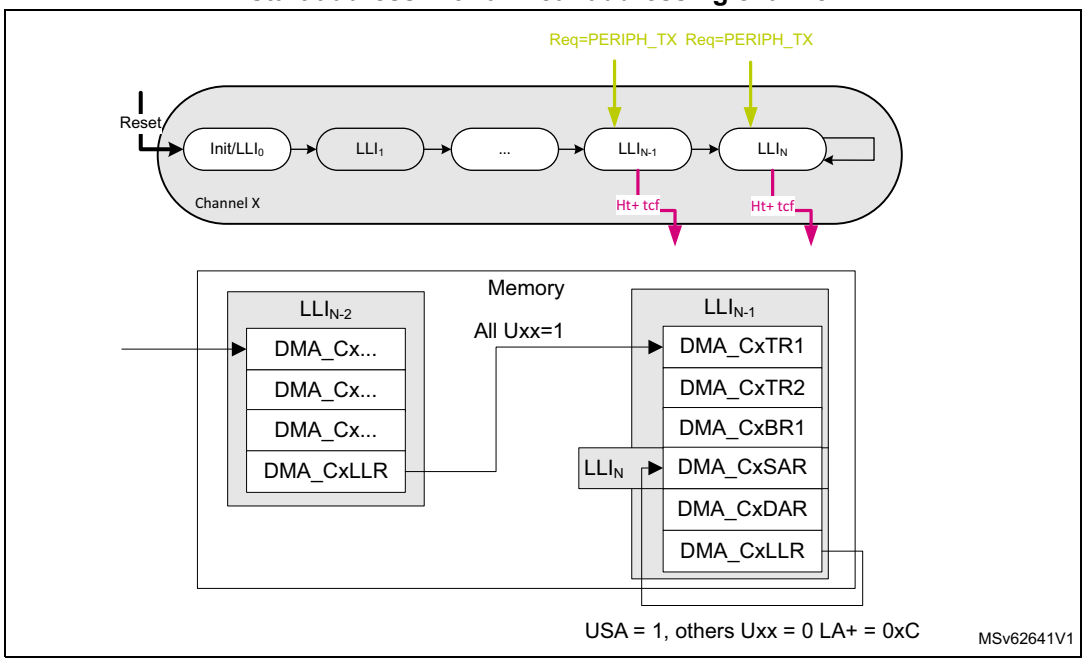


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 60 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 60. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



15.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 15.3.5*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

15.4.15 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

15.4.16 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR, and GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

15.4.17 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 10 to 11): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 10 to 11): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 10 to 11): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])

- for channel x (x = 10 to 11): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
- a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
- a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

15.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

15.6 GPDMA in low-power modes

Table 125. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 15.3.2* to know if any Stop mode is supported.

15.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 126. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 15.4.17*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

15.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

15.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **SECx**: secure state of channel x (x = 11 to 0)
0: Nonsecure
1: Secure

15.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **PRIVx**: privileged state of channel x (x = 11 to 0)
0: unprivileged
1: privileged

15.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR and GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
				rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx and GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 11 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

15.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **MISx**: masked interrupt status of channel x (x = 11 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

15.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **MISx**: masked interrupt status of the secure channel x (x = 11 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

15.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

15.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

15.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level
Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half transfer flag
0: no half transfer event
1: a half transfer event occurred
A half transfer event is either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

15.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 42*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 43*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

15.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see GPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word:
0:no byte-based exchange within the unaligned half-word of each source word
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.
Condition: PAM_1
00: source data is transferred as right aligned, padded with 0s up to the destination data width
01: source data is transferred as right aligned, sign extended up to the destination data width
10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer
Condition: PAM_2
00: source data is transferred as right aligned, left-truncated down to the destination data width
01: source data is transferred as left-aligned, right-truncated down to the destination data width
10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63
The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst
0: fixed burst
1: contiguously incremented burst
The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

15.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	Res.	REQSEL[6:0]						
rw	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (x = 0 to 9), same as 00, channel x (x = 10 to 11), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x = 10 to 11), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 15.3.6*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 10 to 11), for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 9), same as 00; channel x (x = 10 to 11), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 15.3.1 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 15.3.5* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 15.3.3*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[6:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[6:0] is ignored.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **REQSEL[6:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 15.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[6:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

15.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 9)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

15.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 10 to 11)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is

automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]										
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.

– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.

- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

15.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

15.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

15.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 10 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

15.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 10 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRDAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRSAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

15.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 9)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).
Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

15.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 10 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).
Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

15.8.20 GPDMA register map

Table 127. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 127. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x = 0 to 11)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x5C+0x80 * x (x = 0 to 11)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+0x80 * x (x = 0 to 11)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								1
0x64+0x80 * x (x = 0 to 11)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO [1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0						0	0	0
0x90+0x80 * x (x = 0 to 11)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x = 0 to 11)	GPDMA_CxTR2	TCEM [1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM [1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	Res.	REQSEL[6:0]						
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0			0		0	0	0	0	0
0x98+0x80 * x (x = 0 to 9)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x = 10 to 11)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x = 0 to 11)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x = 0 to 11)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x = 10 to 11)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x = 10 to 11)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC+0x80 * x (x = 0 to 9)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xCC+0x80 * x (x = 10 to 11)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

16 Nested vectored interrupt controller (NVIC)

16.1 NVIC main features

- 96 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late arriving interrupts.

The NVIC registers are banked across secure and nonsecure states. All interrupts including the core exceptions are managed by the NVIC. For more information about NVIC and its registers, refer to the *programming manual for Cortex-M33* (PM0223).

16.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers. When TrustZone is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, nonsecure instance

When TrustZone is disabled, only one SysTick timer is available.

The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. In order to match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- When SysTick clock source is CPU clock HCLK reload value = (HCLK x STCALIB) - 1
- When SysTick clock source is external clock (HCLK/8) reload value = ((HCLK/8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.

Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms: SysTick reload value = (100 x STCALIB) - 1 = 0x1869F

For more information about SysTick and its registers, refer to the *programming manual for Cortex-M33* (PM0223).

16.3 Interrupt and exception vectors

The grey rows in the table below indicate core exceptions.

Table 128. STM32U3 series vector table(1)

IRQ number	Priority(2)	Acronym	Description	Address offset
-	-4	Reset	Reset	0x0000 0004
-14	-2	NMI	Non maskable interrupt. The RCC clock security system (CSS) is linked to the NMI vector.	0x0000 0008
-13	-3 or -1	Secure HardFault	Secure hard fault	0x0000 000C
	-1	HardFault	Hard fault	
-12	Settable	MemManage	Memory management	0x0000 0010
-11	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-9	Settable	SecureFault	Secure fault	0x0000 001C
-	-	-	Reserved	0x0000 0020 - 0x0000 0028
-5	Settable	SVC	Supervisor call	0x0000 002C
-4	Settable	DebugMonitor	Debug monitor	0x0000 0030
-	-	-	Reserved	0x0000 0034
-2	Settable	PendSV	Pendable request for system service	0x0000 0038
-1	Settable	SysTick	System tick timer	0x0000 003C
0	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	Settable	PVD_PVM	Programmable voltage detector/peripheral voltage monitor EXTI16/19/20/21/22	0x0000 0044
2	Settable	RTC	RTC nonsecure global interrupts	0x0000 0048
3	Settable	RTC_S	RTC secure global interrupts	0x0000 004C
4	Settable	TAMP	Tamper global interrupts	0x0000 0050
5	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	Settable	FLASH	Flash memory nonsecure global interrupts	0x0000 0058
7	Settable	FLASH_S	Flash memory secure global interrupt	0x0000 005C
8	Settable	GTZC	GTZC1 global interrupt	0x0000 0060
9	Settable	RCC	RCC nonsecure global interrupt	0x0000 0064

Table 128. STM32U3 series vector table(1) (continued)

IRQ number	Priority(2)	Acronym	Description	Address offset
10	Settable	RCC_S	RCC secure global interrupt	0x0000 0068
11	Settable	EXTI0	EXTI line0 interrupt	0x0000 006C
12	Settable	EXTI1	EXTI line1 interrupt	0x0000 0070
13	Settable	EXTI2	EXTI line2 interrupt	0x0000 0074
14	Settable	EXTI3	EXTI line3 interrupt	0x0000 0078
15	Settable	EXTI4	EXTI line4 interrupt	0x0000 007C
16	Settable	EXTI5	EXTI line5 interrupt	0x0000 0080
17	Settable	EXTI6	EXTI line6 interrupt	0x0000 0084
18	Settable	EXTI7	EXTI line7 interrupt	0x0000 0088
19	Settable	EXTI8	EXTI line8 interrupt	0x0000 008C
20	Settable	EXTI9	EXTI line9 interrupt	0x0000 0090
21	Settable	EXTI10	EXTI line10 interrupt	0x0000 0094
22	Settable	EXTI11	EXTI line11 interrupt	0x0000 0098
23	Settable	EXTI12	EXTI line12 interrupt	0x0000 009C
24	Settable	EXTI13	EXTI line13 interrupt	0x0000 00A0
25	Settable	EXTI14	EXTI line14 interrupt	0x0000 00A4
26	Settable	EXTI15	EXTI line15 interrupt	0x0000 00A8
27	Settable	IWDG	Independent watchdog interrupt	0x0000 00AC
28	Settable	SAES	Secure AES	0x0000 00B0
29	Settable	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00B4
30	Settable	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00B8
31	Settable	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00BC
32	Settable	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00C0
33	Settable	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00C4
34	Settable	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00C8
35	Settable	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00CC
36	Settable	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00D0
37	Settable	ADC1	ADC1 global interrupt	0x0000 00D4
38	Settable	DAC1	DAC1 global interrupt	0x0000 00D8

Table 128. STM32U3 series vector table(1) (continued)

IRQ number	Priority(2)	Acronym	Description	Address offset
39	Settable	FDCAN1_IT0	FDCAN1 interrupt 0	0x0000 00DC
40	Settable	FDCAN1_IT1	FDCAN1 interrupt 1	0x0000 00E0
41	Settable	TIM1_BRK TIM1_TERR TIM1_IERR	TIM1 break TIM1 transition error TIM1 index error	0x0000 00E4
42	Settable	TIM1_UP	TIM1 update	0x0000 00E8
43	Settable	TIM1_TRG_COM TIM1_DIR TIM1_IDX	TIM1 trigger and commutation TIM1 direction change interrupt TIM1 index	0x0000 00EC
44	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0
45	Settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	Settable	TIM3	TIM3 global interrupt	0x0000 00F8
47	Settable	TIM4	TIM4 global interrupt	0x0000 00FC
48	-	-	Reserved	0x0000 0100
49	Settable	TIM6	TIM6 global interrupt	0x0000 0104
50	Settable	TIM7	TIM7 global interrupt	0x0000 0108
51	-	-	Reserved	0x0000 010C
52	-	-	Reserved	0x0000 0110
53	Settable	I3C1_EV	I3C1 event interrupt	0x0000 0114
54	Settable	I3C1_ER	I3C1 error interrupt	0x0000 0118
55	Settable	I2C1_EV	I2C1 event interrupt	0x0000 011C
56	Settable	I2C_ER	I2C1 error interrupt	0x0000 0120
57	Settable	I2C2_EV	I2C2 event interrupt	0x0000 0124
58	Settable	I2C2_ER	I2C2 error interrupt	0x0000 0128
59	Settable	SPI1	SPI1 global interrupt	0x0000 012C
60	Settable	SPI2	SPI2 global interrupt	0x0000 0130
61	Settable	USART1	USART1 global interrupt	0x0000 0134
62	-	-	Reserved	0x0000 0138
63	Settable	USART3	USART3 global interrupt	0x0000 013C
64	Settable	UART4	UART4 global interrupt	0x0000 0140

Table 128. STM32U3 series vector table(1) (continued)

IRQ number	Priority(2)	Acronym	Description	Address offset
65	Settable	UART5	UART5 global interrupt	0x0000 0144
66	Settable	LPUART1	LPUART1 global interrupt	0x0000 0148
67	Settable	LPTIM1	LPTIM1 global interrupt	0x0000 014C
68	Settable	LPTIM2	LPTIM2 global interrupt	0x0000 0150
69	Settable	TIM15	TIM15 global interrupt	0x0000 0154
70	Settable	TIM16	TIM16 global interrupt	0x0000 0158
71	Settable	TIM17	TIM17 global interrupt	0x0000 015C
72	Settable	COMP	COMP1/COMP2	0x0000 0160
73	Settable	USB	USB global interrupt	0x0000 0164
74	Settable	CRS	Clock recovery system global interrupt	0x0000 0168
75	-	-	Reserved	0x0000 016C
76	Settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 0170
77	-	-	Reserved	0x0000 0174
78	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 0178
79	-	-	Reserved	0x0000 017C
80	Settable	GPDMA1_CH8	GPDMA1 channel 8 interrupt	0x0000 0180
81	Settable	GPDMA1_CH9	GPDMA1 channel 9 interrupt	0x0000 0184
82	Settable	GPDMA1_CH10	GPDMA1 channel 10 interrupt	0x0000 0188
83	Settable	GPDMA1_CH11	GPDMA1 channel 11 interrupt	0x0000 018C
84	-	-	Reserved	0x0000 0190
85	-	-	Reserved	0x0000 0194
86	-	-	Reserved	0x0000 0198
87	-	-	Reserved	0x0000 019C
88	Settable	I2C3_EV	I2C3 event interrupt	0x0000 01A0
89	Settable	I2C3_ER	I2C3 error interrupt	0x0000 01A4
90	Settable	SAI1	SAI1 global interrupt	0x0000 01A8
91	-	-	Reserved	0x0000 01AC
92	Settable	TSC	TSC global interrupt	0x0000 01B0
93	Settable	AES	AES global interrupt	0x0000 01B4

Table 128. STM32U3 series vector table(1) (continued)

IRQ number	Priority(2)	Acronym	Description	Address offset
94	Settable	RNG	RNG global interrupt	0x0000 01B8
95	Settable	FPU	Floating point interrupt	0x0000 01BC
96	Settable	HASH	HASH interrupt	0x0000 01C0
97	Settable	PKA	PKA global interrupt	0x0000 01C4
98	Settable	LPTIM3	LPTIM3 global interrupt	0x0000 01C8
99	Settable	SPI3	SPI3 global interrupt	0x0000 01CC
100	Settable	I3C2_EV	I3C2 event interrupt	0x0000 01D0
101	Settable	I3C2_ER	I3C2 error interrupt	0x0000 01D4
102	-	-	Reserved	0x0000 01D8
103	-	-	Reserved	0x0000 01DC
104	-	-	Reserved	0x0000 01E0
105	-	-	Reserved	0x0000 01E4
106	-	-	Reserved	0x0000 01E8
107	Settable	ICACHE	Instruction cache global interrupt	0x0000 01EC
108	-	-	Reserved	0x0000 01F0
109	-	-	Reserved	0x0000 01F4
110	Settable	LPTIM4	LPTIM4 global interrupt	0x0000 01EC
111	-	-	Reserved	0x0000 01FC
112	Settable	ADF1	ADF1 interrupt	0x0000 0200
113	Settable	ADC2	ADC2 global interrupt	0x0000 0204
114	-	-	Reserved	0x0000 0208
115	-	-	Reserved	0x0000 020C
116	-	-	Reserved	0x0000 0210
117	-	-	Reserved	0x0000 0214
118	-	-	Reserved	0x0000 0218
119	-	-	Reserved	0x0000 021C
120	-	-	Reserved	0x0000 0220
121	-	-	Reserved	0x0000 0224
122	-	-	Reserved	0x0000 0228

Table 128. STM32U3 series vector table(1) (continued)

IRQ number	Priority(2)	Acronym	Description	Address offset
123	Settable	PWR	PWR nonsecure interrupt	0x0000 022C
124	Settable	PWR_S	PWR secure interrupt	0x0000 0230

1. Some interrupt lines are only available on some STM32U3 series devices. Refer to the device datasheet for availability of associated peripheral. If not present, consider this interrupt line as reserved.
2. If multiple pending interrupts have the same programmed priority level, the pending interrupt with the lowest IRQ number takes precedence. For example, if both interrupt[n] and interrupt[n+1] are pending and have the same priority, then interrupt[n] is processed before interrupt[n+1]. For further information, refer to the Arm Cortex-M33 technical documentation.

17 Extended interrupts and event controller (EXTI)

17.1 EXTI introduction

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable event inputs. It provides wake-up requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes. The EXTI also includes the EXTI mux I/O port selection.

17.2 EXTI main features

The EXTI main features are the following:

- Up to 23 input events supported
- All event inputs allow the possibility to wake up the system
- Events that do not have an associated wake-up flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI
- Events can be used to generate a CPU wake-up event
- Secure events: The access to control and configuration bits of secure input events can be made secure
- EXTI I/O port selection

The configurable events have the following features:

- Selectable active trigger edge
- Interrupt pending status register bits independent for the rising and falling edge
- Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
- Software trigger possibility

17.3 EXTI functional description

17.3.1 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux as shown in *Figure 61*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and their masking.

The EXTI mux provides the I/O port selection of the EXTI event signal.

Figure 61. EXTI block diagram

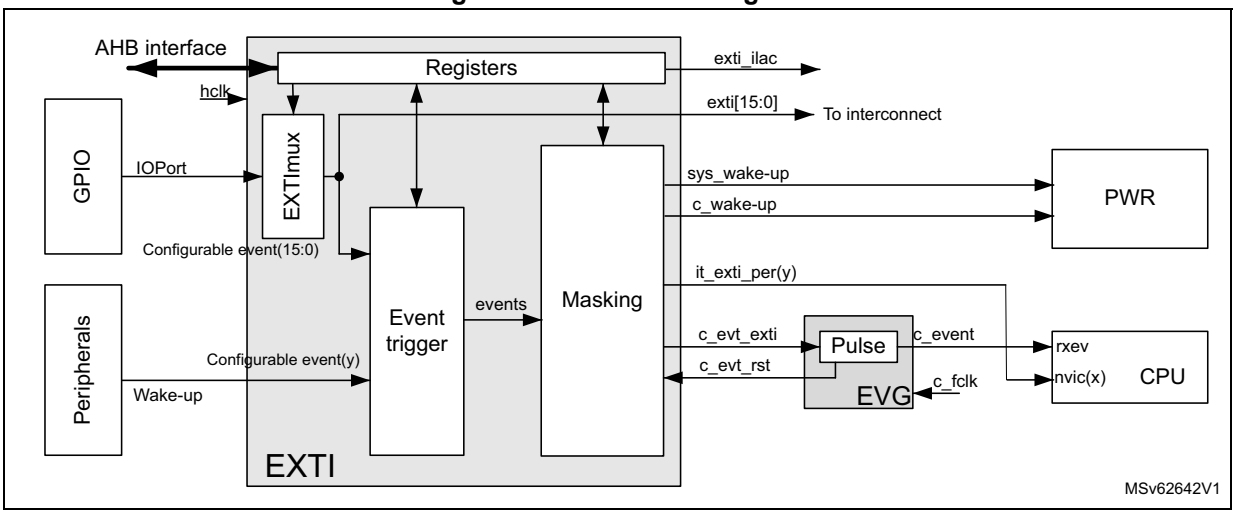


Table 129. EXTI signals

Name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up events from peripherals that do not have an associated interrupt and flag in the peripheral
exti_ilac	O	Illegal access event
IOPort(n)	I	GPIOs block I/O ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 130. EVG signals

Name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit that requires to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- All GPIO ports input to the EXTI multiplexer allow the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR and are used to wake up the system and the CPU sub- system bus clocks.

EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

Table 131. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17	COMP1 output	Configurable
18	COMP2 output	Configurable
19	V_{DDUSB} voltage monitor	Configurable
20	V_{DDIO2} voltage monitor	Configurable
21	V_{DDA} voltage monitor 1	Configurable
22	V_{DDA} voltage monitor 2	Configurable

17.3.2 Event features control

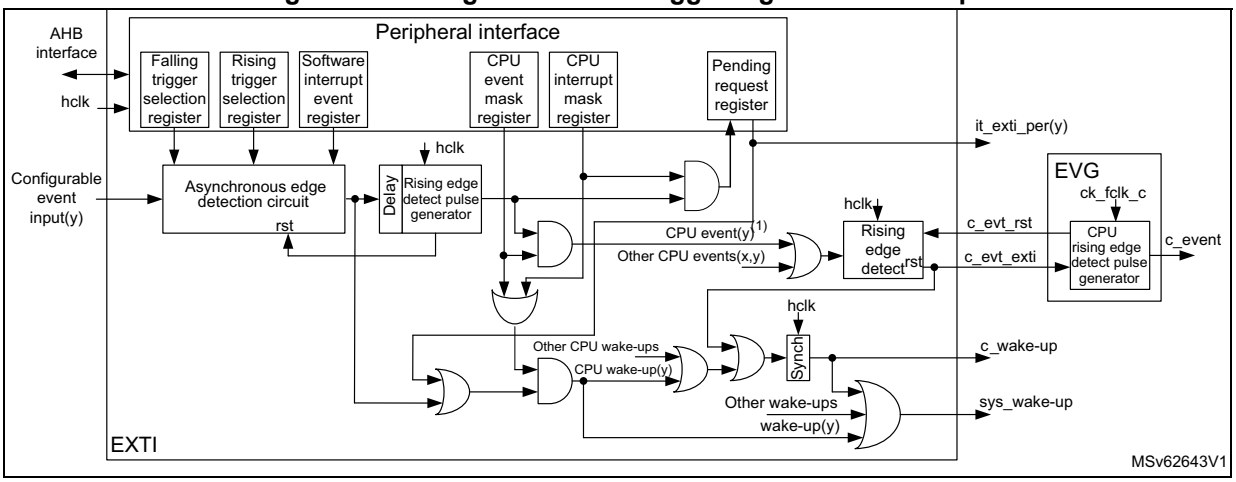
The events features are controlled from register bits as follows:

- Active trigger edge enable by
  - rising edge selection in the *EXTI rising trigger selection register (EXTI_RTSR1)*
  - falling edge selection in the *EXTI falling trigger selection register (EXTI_FTSR1)*
- Software trigger in the *EXTI software interrupt event register (EXTI_SWIER1)*
- Interrupt pending flag in the
  - *EXTI rising edge pending register (EXTI_RPR1)*
  - *EXTI falling edge pending register (EXTI_FPR1)*
- CPU wake-up and interrupt enable in the
  - *EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)*
- CPU wake-up and event enable
  - *EXTI CPU wake-up with event mask register (EXTI_EMR1)*

17.3.3 EXTI configurable event input wake-up

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wake-up event.

Figure 62. Configurable event trigger logic CPU wake-up



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR1 and EXTI_FPR1) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR1 and/or EXTI_FPR1 registers.

When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

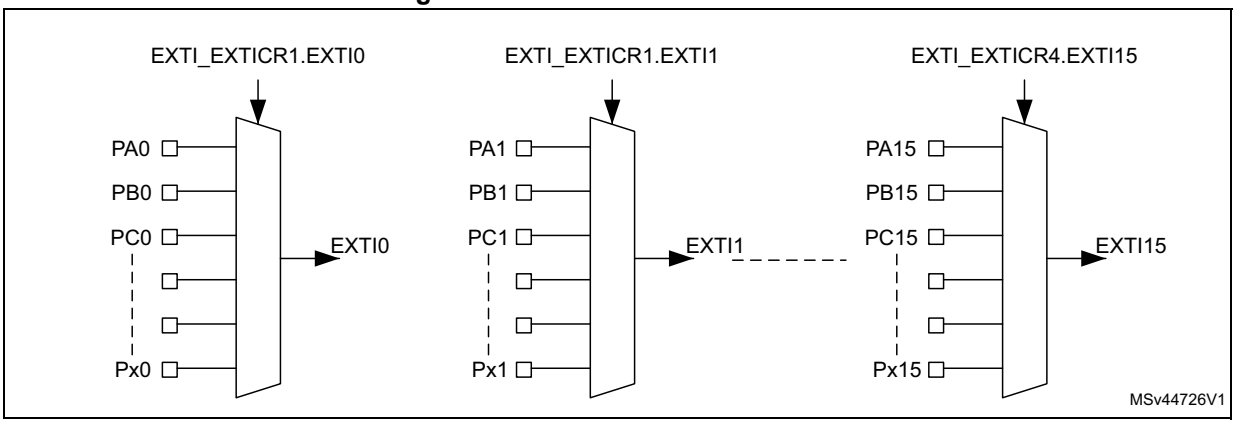
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

17.3.4 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event.

The selection of GPIO port as EXTI mux output is controlled in the EXTI_EXTICRx.

Figure 63. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR1 and EXTI_EMR1 registers.

17.3.5 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 132. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)	Exti(n) interrupt(1)	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. 2.Only if CPU interrupt is enabled in EXTI_IMR1.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR1.RPIFn and/or EXTI_FPR1.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR1.RPIFn and/or EXTI_FPR1.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER1, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR1, whatever the setting in EXTI_RTSR1.

17.3.6 EXTI event protection

The EXTI is able to protect event register bits from being modified by nonsecure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFGR1 and EXTI_PRIVCFGR1. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

Table 133. Register protection overview

Register name	Access type	Protection(1)(2)
EXTI_RTSR1	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR1 and EXTI_PRIVCFGR1.
EXTI_FTSR1	RW	
EXTI_SWIER1	RW	
EXTI_RPR1	RW	
EXTI_FPR1	RW	
EXTI_SECCFGR1	RW	Always secure. Privilege can be bit-wise enabled in EXTI_PRIVCFGR1.
EXTI_PRIVCFGR1	RW	Always privilege. Security can be bit-wise enabled in EXTI_SECCFGR1.
EXTI_EXTICRx	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR1 and EXTI_PRIVCFGR1.
EXTI_LOCKR	RW	Always secure.
EXTI_IMR1	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR1 and EXTI_PRIVCFGR1.
EXTI_EMR1	RW	

1. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR1 register).
2. Security is enabled with the individual input event (EXTI_SECCFGR1 register).

EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access. A nonsecure write access is discarded and a read returns 0.

When input events are nonsecure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and nonsecure access.

The security configuration in EXTI_SECCFGR1 can be globally locked after reset by EXTI_LOCKR.LOCK.

EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

The privileged configuration in EXTI_PRIVCFGR1 can be globally locked after reset by EXTI_LOCKR.LOCK.

17.4 EXTI registers

All registers can be accessed with word (32-bit), half-word (16-bit), and byte (8-bit) access.

17.4.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	RT18	RT17	RT16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **RTi:** Rising trigger event configuration bit of configurable event input i[(1)] (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, RTx can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, RTi can only be accessed with secure access. Nonsecure write to this bit i is discarded and nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, RTi can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, RTi can only be accessed with privileged access. Unprivileged write to this bit i is discarded. Unprivileged read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.4.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	FT18	FT17	FT16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **FTi:** Falling trigger event configuration bit of configurable event input i[(1)] (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, FTi can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, FTi can only be accessed with secure access. Nonsecure write to this FTi is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, FTi can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, FTi can only be accessed with privileged access. Unprivileged write to this FTi is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line.

1. The configurable event inputs are edge triggered. No glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

17.4.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **SWIi:** Software interrupt on event i (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, SWIi can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, SWIi can only be accessed with secure access. Nonsecure write to this SWIi is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, SWIi can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, SWIi can only be accessed with privileged access. Unprivileged write to this SWIi is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event i. This bit is auto cleared by hardware.

17.4.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF22	RPIF21	RPIF20	RPIF19	RPIF18	RPIF17	RPIF16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **RPIFi:** Configurable event input i rising edge pending bit (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, RPIFi can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, RPIFi can only be accessed with secure access. Nonsecure write to this RPIFi is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, RPIFi can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, RPIFi can only be accessed with privileged access. Unprivileged write to this RPIFi is discarded, unprivileged read returns 0.

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

0: No rising edge trigger request occurred.

1: Rising edge trigger request occurred.

17.4.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF22	FPIF21	FPIF20	FPIF19	FPIF18	FPIF17	FPIF16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **FPIFi:** configurable event inputs i falling edge pending bit (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, FPIFi can be accessed with nonsecure and secure access.
When EXTI_SECCFGR.SECi is enabled, FPIFi can only be accessed with secure access. Nonsecure write to this FPIFi is discarded, nonsecure read returns 0.
When EXTI_PRIVCFGR.PRIVi is disabled, FPIFi can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVi is enabled, FPIFi can only be accessed with privileged access. Unprivileged write to this FPIFi is discarded, unprivileged read returns 0.
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred

17.4.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security. A nonsecure write access is ignored and causes the generation of an illegal access event. A nonsecure read returns the register data. This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **SECi:** Security enable on event input i (i = 22 to 0)
When EXTI_PRIVCFGR.PRIVi is disabled, SECi can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVi is enabled, SECi can only be written with privileged access. Unprivileged write to this SECi is discarded.
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

17.4.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data. This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **PRIVi:** Security enable on event input i (i = 22 to 0)
When EXTI_SECCFGR.SECi is disabled, PRIVi can be accessed with secure and nonsecure access.
When EXTI_SECCFGR.SECi is enabled, PRIVi can only be written with secure access. Nonsecure write to this PRIVi is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

17.4.8 EXTI external interrupt selection register (EXTI_EXTICRx)

Address offset: 0x060 + 0x4 * (x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI{4*(x-1)+3}[7:0]								EXTI{4*(x-1)+2}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI{4*(x-1)+1}[7:0]								EXTI{4*(x-1)}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI{4 * (x - 1) + 3}[7:0]:** EXTI{4 * (x - 1) + 3} GPIO port selection

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 3} external interrupt.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 3} is disabled, this bitfield can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 3} is enabled, this bitfield can only be accessed with secure access. Nonsecure write is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 3} is disabled, this bitfield can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 3} is enabled, this bitfield can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA[{4 * (x - 1) + 3}] pin
0x01: PB[{4 * (x - 1) + 3}] pin
0x02: PC[{4 * (x - 1) + 3}] pin
0x03: PD[{4 * (x - 1) + 3}] pin
0x04: PE[{4 * (x - 1) + 3}] pin
0x06: PG[{4 * (x - 1) + 3}] pin
0x07: PH[{4 * (x - 1) + 3}] pin
Others: Reserved

Bits 23:16 **EXTI{4 * (x - 1) + 2}[7:0]:** EXTI{4 * (x - 1) + 2} GPIO port selection

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 2} external interrupt.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 2} is disabled, this bitfield can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 2} is enabled, this bitfield can only be accessed with secure access. Nonsecure write is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 2} is disabled, this bitfield can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 2} is enabled, this bitfield can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA[{4 * (x - 1) + 2}] pin
0x01: PB[{4 * (x - 1) + 2}] pin
0x02: PC[{4 * (x - 1) + 2}] pin
0x03: PD[{4 * (x - 1) + 2}] pin
0x04: PE[{4 * (x - 1) + 2}] pin
0x06: PG[{4 * (x - 1) + 2}] pin
0x07: PH[{4 * (x - 1) + 2}] pin
Others: Reserved

Bits 15:8 **EXTI{4 * (x - 1) + 1}[7:0]:** EXTI{4 * (x - 1) + 1} GPIO port selection

These bits are written by software to select the source input for EXTI{4 * (x - 1) + 1} external interrupt.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 1} is disabled, this bitfield can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SEC{4 * (x - 1) + 1} is enabled, this bitfield can only be accessed with secure access. Nonsecure write is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 1} is disabled, this bitfield can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1) + 1} is enabled, this bitfield can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA[{4 * (x - 1) + 1}] pin
0x01: PB[{4 * (x - 1) + 1}] pin
0x02: PC[{4 * (x - 1) + 1}] pin
0x03: PD[{4 * (x - 1) + 1}] pin
0x04: PE[{4 * (x - 1) + 1}] pin
0x06: PG[{4 * (x - 1) + 1}] pin
0x07: PH[{4 * (x - 1) + 1}] pin
Others: Reserved

Bits 7:0 **EXTI{4 * (x - 1)}[7:0]:** EXTI{4 * (x - 1)} GPIO port selection

These bits are written by software to select the source input for EXTI{4 * (x - 1)} external interrupt.

When EXTI_SECCFGR.SEC{4 * (x - 1)} is disabled, this bitfield can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SEC{4 * (x - 1)} is enabled, this bitfield can only be accessed with secure access. Nonsecure write is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1)} is disabled, this bitfield can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIV{4 * (x - 1)} is enabled, this bitfield can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA[{4 * (x - 1)}] pin
0x01: PB[{4 * (x - 1)}] pin
0x02: PC[{4 * (x - 1)}] pin
0x03: PD[{4 * (x - 1)}] pin
0x04: PE[{4 * (x - 1)}] pin
0x06: PG[{4 * (x - 1)}] pin
0x07: PH[{4 * (x - 1)}] pin
Others: Reserved

17.4.9 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides write access security: a nonsecure write access is ignored, a read access returns zero data, and both generates an illegal access event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **LOCK**: Global security and privilege configuration registers (EXTI_SECCFGR and EXTI_PRIVCFGR) lock

This bit is written once after reset.

0: Security and privilege configuration open, can be modified.

1: Security and privilege configuration locked, can no longer be modified.

17.4.10 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x0000 0000

This register contains bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM22	IM21	IM20	IM19	IM18	IM17	IM16
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **IMi:** CPU wake-up with interrupt mask on event input i[(1)] (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, IMi can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, IMi can only be accessed with secure access. Nonsecure write to this bit is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, IMi can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, IMi can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event i is masked.

1: Wake-up with interrupt request from input event i is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

17.4.11 EXTI CPU wake-up with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM22	EM21	EM20	EM19	EM18	EM17	EM16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **EMi:** CPU wake-up with event generation mask on event input i (i = 22 to 0)

When EXTI_SECCFGR.SECi is disabled, EMi can be accessed with nonsecure and secure access.

When EXTI_SECCFGR.SECi is enabled, EMi can only be accessed with secure access. Nonsecure write to this EMi is discarded. Nonsecure read returns 0.

When EXTI_PRIVCFGR.PRIVi is disabled, EMi can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVi is enabled, EMi can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from line i is masked.

1: Wake-up with event generation from line i is unmasked.

17.4.12 EXTI register map

Table 134. EXTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	RT18	RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	FT18	FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI22	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF22	RPIF21	RPIF20	RPIF19	RPIF18	RPIF17	RPIF16	RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF22	FPIF21	FPIF20	FPIF19	FPIF18	FPIF17	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	EXTI_SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	EXTI_PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x05C	Reserved	Reserved																															

Table 134. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	EXTI_EXTICR1	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	EXTI_EXTICR3	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	EXTI_EXTICR4	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI2[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	EXTI_LOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x074-0x07C	Reserved	Reserved																															
0x080	EXTI_IMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

18 Cyclic redundancy check calculation unit (CRC)

18.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

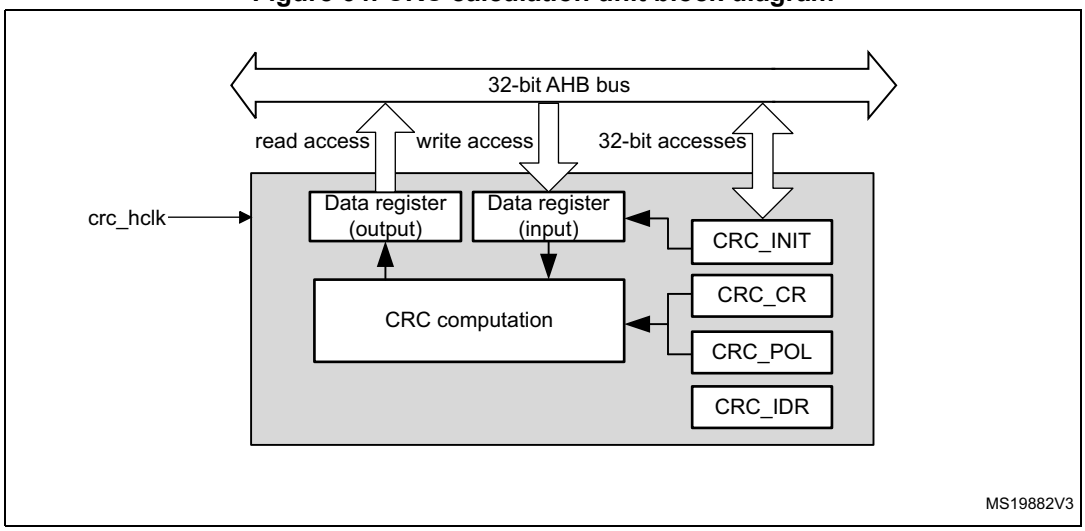
18.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

18.3 CRC functional description

18.3.1 CRC block diagram

Figure 64. CRC calculation unit block diagram



18.3.2 CRC internal signals

Table 135. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

18.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

18.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

18.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

18.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

18.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
					rw	rw	rw	rw	rw	rw	rw	rw			rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size
These bits control the size of the polynomial.
00: 32 bit polynomial
01: 16 bit polynomial
10: 8 bit polynomial
11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit
This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

18.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value
This register is used to write the CRC initial value.

18.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

18.4.6 CRC register map

Table 136. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT[1]	RTYPE_IN[1]	REV_OUT[1]	REV_OUT[0]	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

19 Extended-SPI interface (OCTOSPI)

19.1 OCTOSPI introduction

The OCTOSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the OCTOSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The OCTOSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

19.2 OCTOSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, and octal communication
- Dual memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AHB interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.

19.3 OCTOSPI implementation

Table 137. OCTOSPI implementation

OCTOSPI feature	OCTOSPI1
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
Dual-octal configuration	-
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS, DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Compliant with dual-OCTOSPI arbiter (communication regulation)	-
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
GPDMA interface	X
High-speed interface	-
Dual chip select	-
Extended external memory	-
Prefetch disable	-
Prefetch hardware software	-

19.4 OCTOSPI functional description

19.4.1 OCTOSPI block diagram

Figure 65. OCTOSPI block diagram in octal configuration

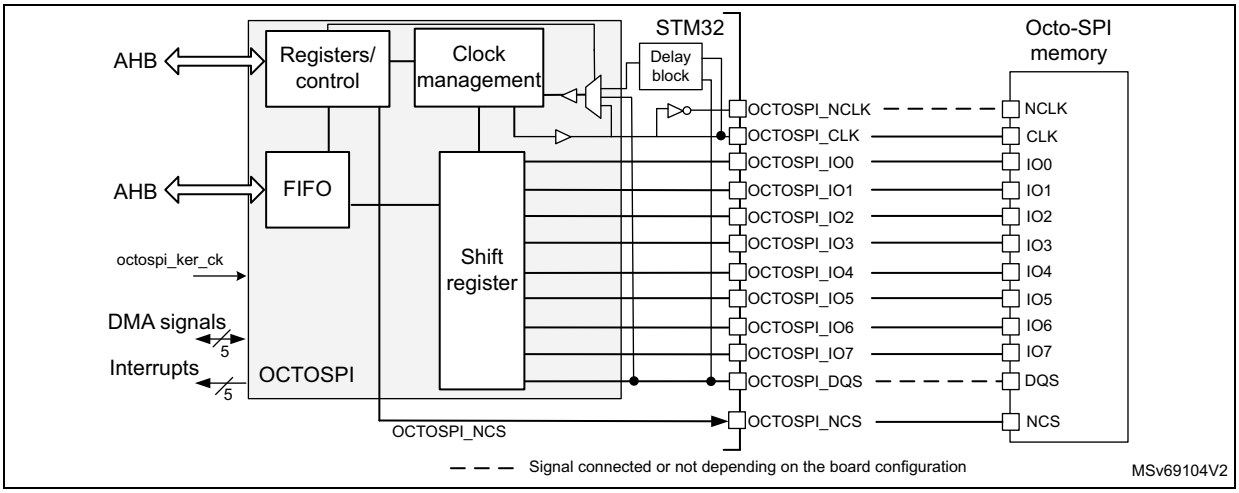


Figure 66. OCTOSPI block diagram in quad configuration

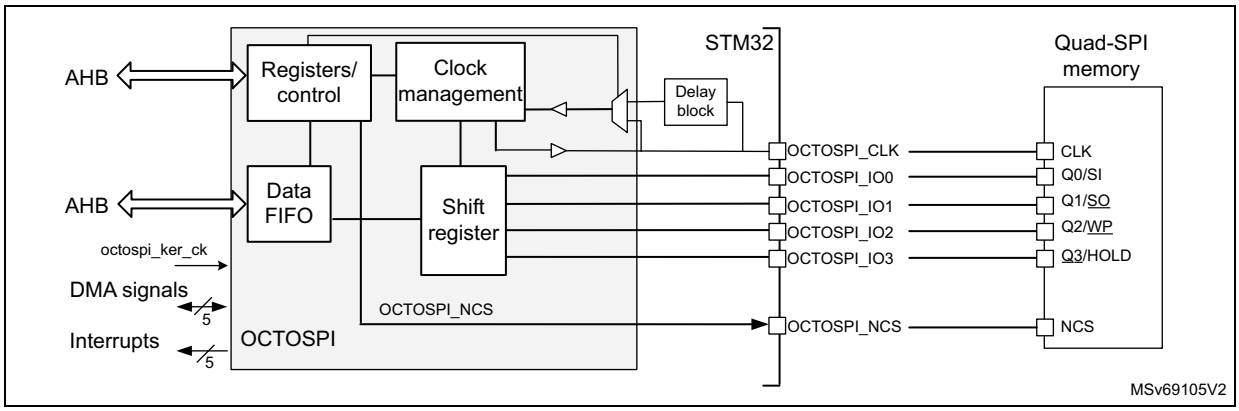
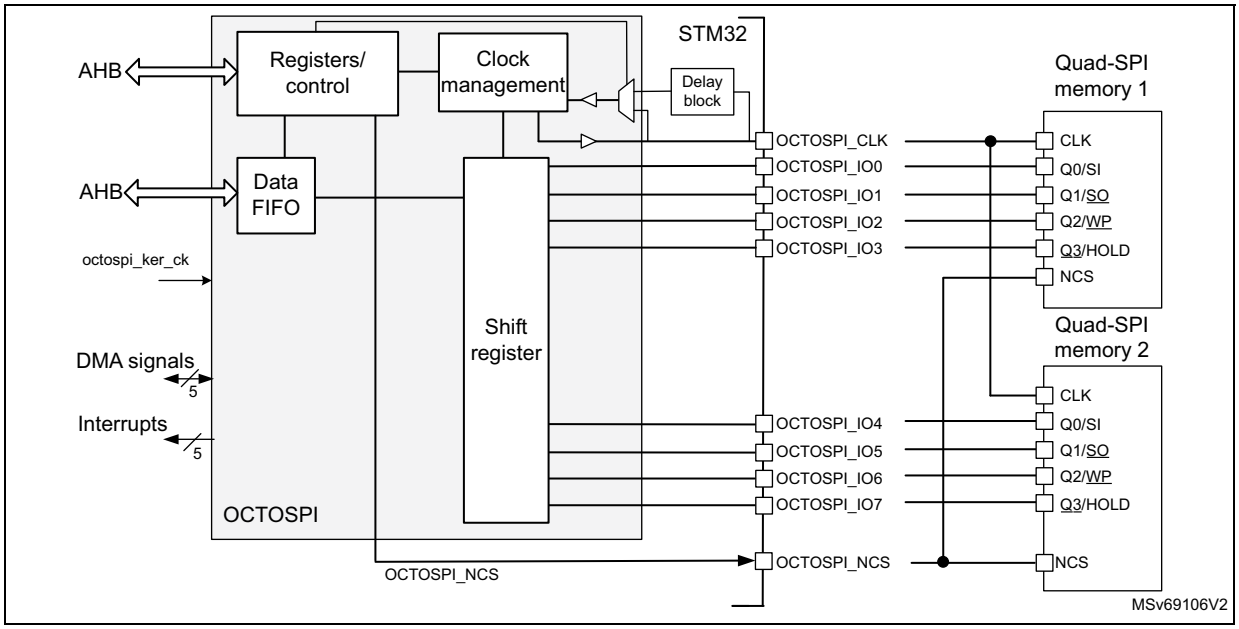


Figure 67. OCTOSPI block diagram in dual-quad configuration



19.4.2 OCTOSPI pins and internal signals

Table 138. OCTOSPI input/output pins

Pin name	Type	Description
OCTOSPI_NCLK	Output	OCTOSPI inverted clock to support 1.8 V HyperBus protocol
OCTOSPI_CLK		OCTOSPI clock
OCTOSPI_IOn (n = 0 to 7)	Input/output	OCTOSPI data pins
OCTOSPI_NCS	Output	Chip select for the memory
OCTOSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all OCTOSPI input/output pins to avoid any data corruption.

Table 139. OCTOSPI internal signals

Signal name	Type	Description
octospi_hclk	Input	OCTOSPI AHB clock
octospi_ker_ck	Input	OCTOSPI kernel clock
octospi_dma	N/A	DMA request signal
octospi_it	Output	Global interrupt line (see *Table 143* for the multiple sources of interrupt)

a. When applicable.

19.4.3 OCTOSPI interface to memory modes

The OCTOSPI supports the following protocols:

- regular-command protocol
- HyperBus protocol

The OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

19.4.4 OCTOSPI regular-command protocol

When in regular-command protocol, the OCTOSPI communicates with the external device using commands. Each command can include the following phases:

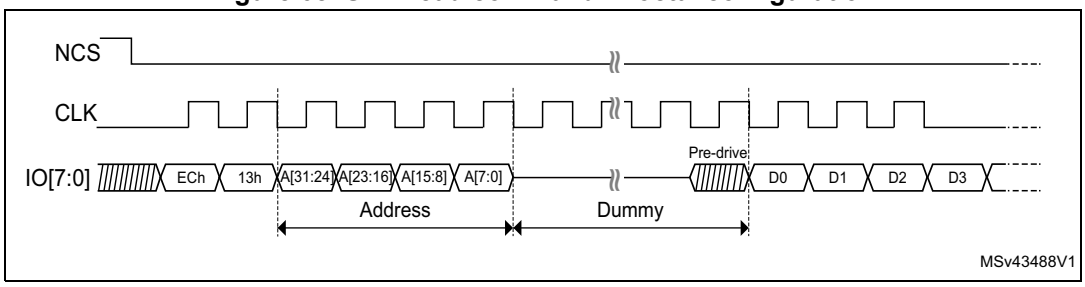
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 68. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in OCTOSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of OCTOSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of OCTOSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in OCTOSPI_WIR, and the instruction format is specified in OCTOSPI_WCCR. The instruction used for the read operation and the instruction format are specified in OCTOSPI_IR and OCTOSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in OCTOSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in OCTOSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in OCTOSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in OCTOSPI_WCCR. The address format for the read operation is specified in OCTOSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in OCTOSPI_CCR. The bytes to be sent are specified in OCTOSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in OCTOSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in OCTOSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in OCTOSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in OCTOSPI_WABR, and the alternate byte format is specified in OCTOSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in OCTOSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-SPI, the quad-SPI, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in OCTOSPI_WTCR. The dummy cycles for the read operation are specified in OCTOSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in OCTOSPI_DLR. In this mode, the data to be sent to the external device must be written to OCTOSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading OCTOSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in OCTOSPI_DLR, and the data received from the external device can be obtained by reading OCTOSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in OCTOSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

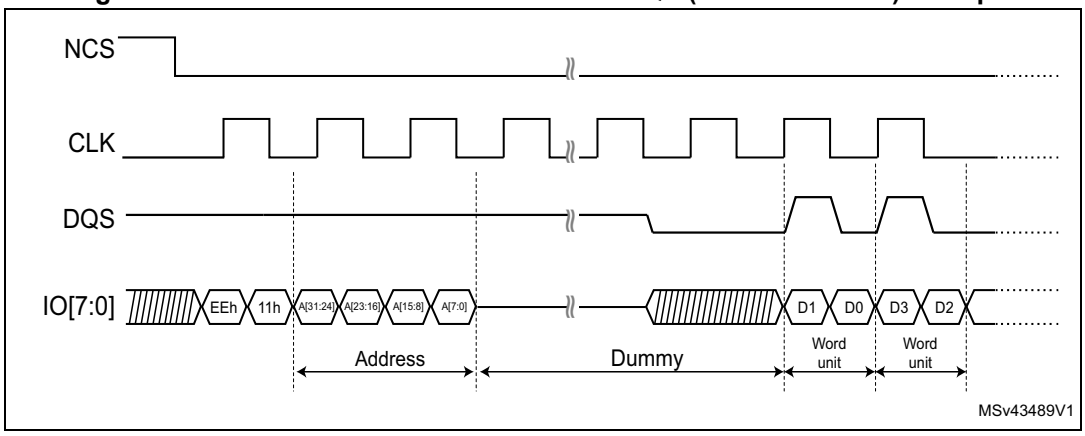
In memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in OCTOSPI_CCR.

Figure 69. DTR read in octal-SPI mode with DQS (Macronix mode) example



19.4.5 OCTOSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the OCTOSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the “write protect” function).
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, IO0/IO1 are always high-impedance.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0 to IO3 are all high-impedance.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

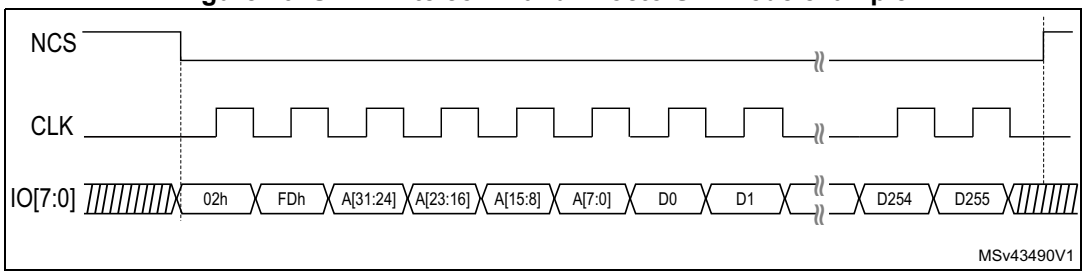
Single-data rate (SDR) mode

By default, all the phases operate in SDR mode.

In this mode, when the OCTOSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 70. SDR write command in octo-SPI mode example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in OCTOSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in OCTOSPI_WCCR. The DTR mode for each phase of the read operations is specified in OCTOSPI_CCR.

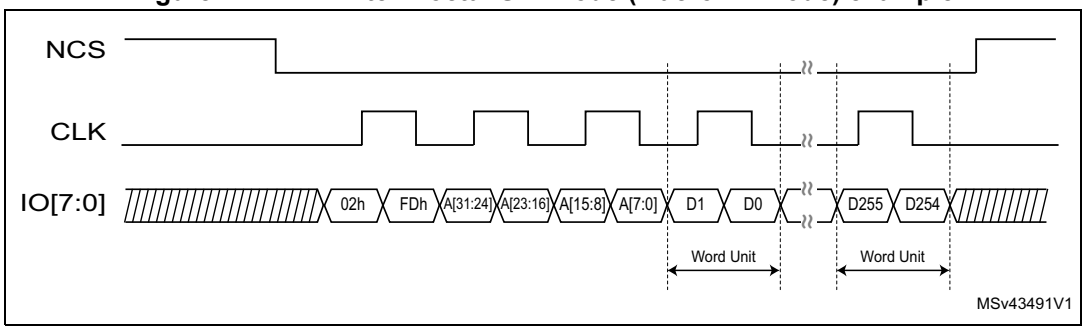
In DTR mode, when the OCTOSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 71. DTR write in octal-SPI mode (Macronix mode) example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Dual-quad configuration

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 011, two external Quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in OCTOSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in OCTOSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

19.4.6 HyperBus protocol

The OCTOSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

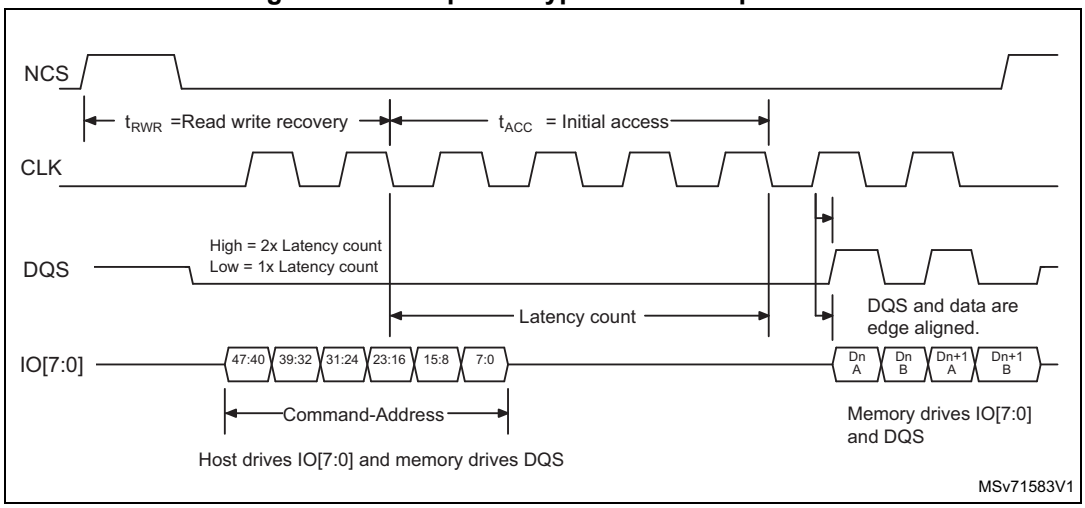
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 72. Example of HyperBus read operation



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 140. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in OCTOSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of OCTOSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in OCTOSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in OCTOSPI_HLCR) according to the memory latency

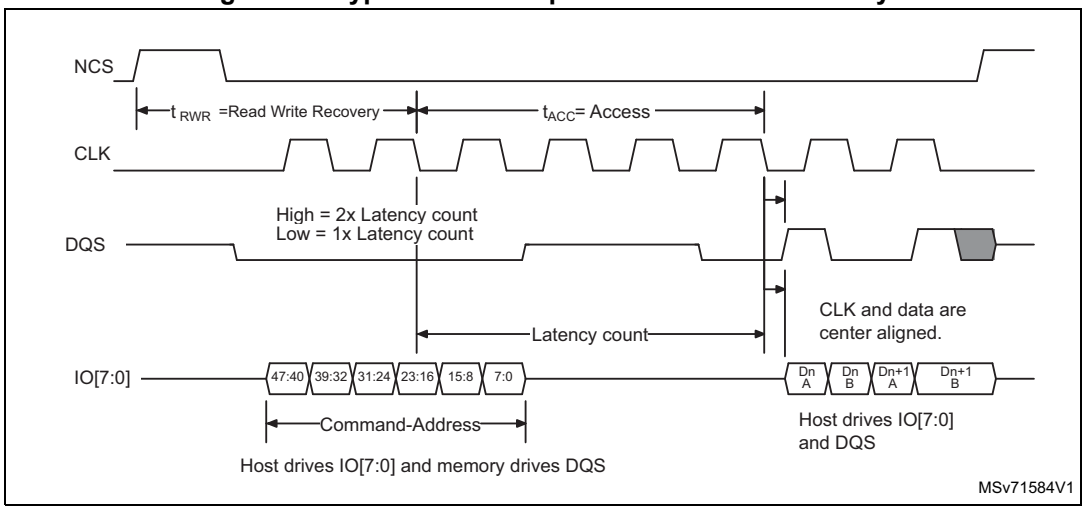
During the read operation, the DQS is used by the device, in two ways (see *Figure 72*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the OCTOSPI, during the data phase, for write data masking.

Figure 73. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the OCTOSPI to meet the device request.

Figure 74. HyperBus read operation with additional latency

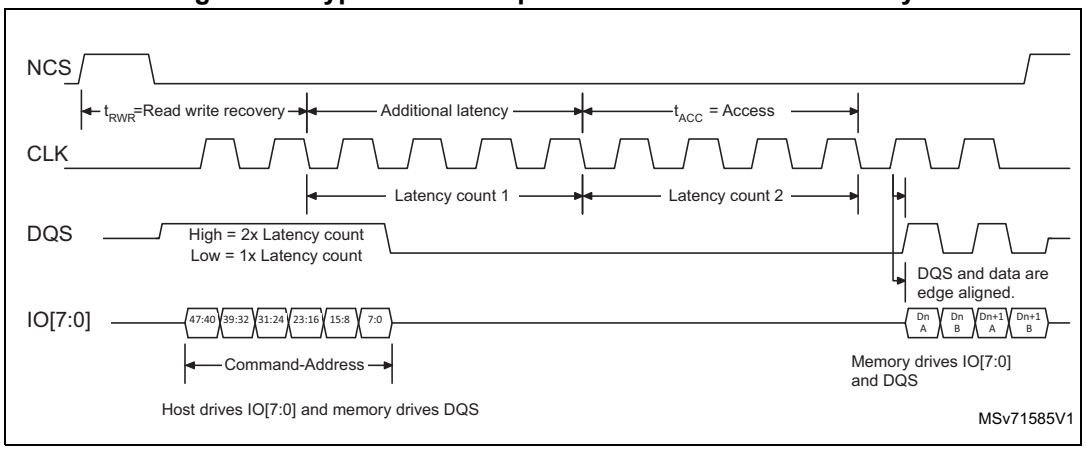
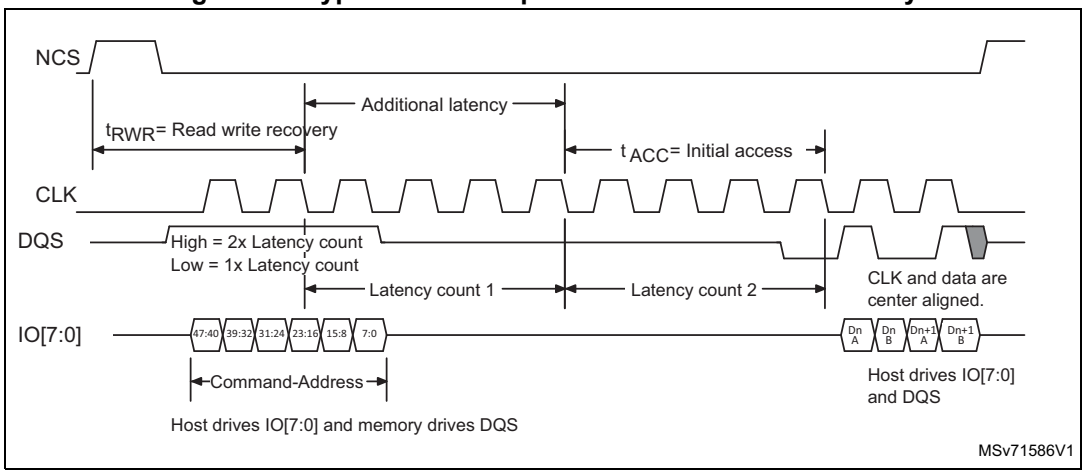


Figure 75. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

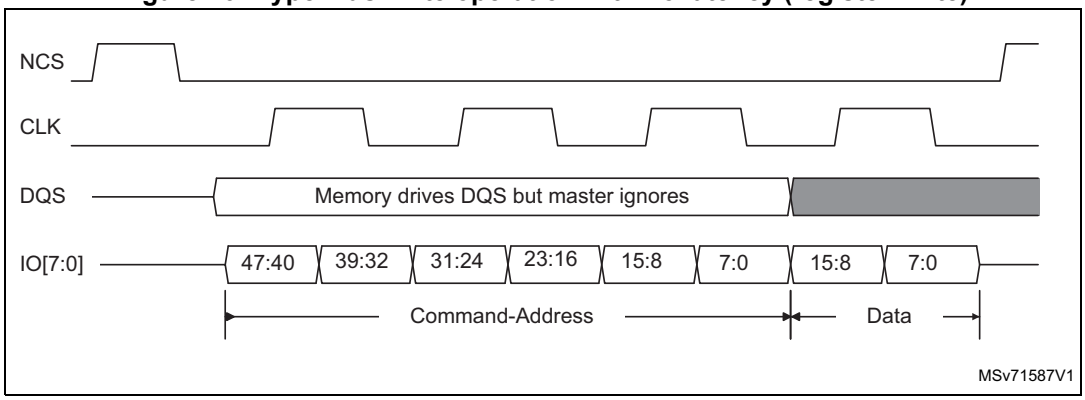
The latency can be forced to 2 x t_{ACC} by setting LM in OCTOSPI_HLCR.

In this OCTOSPI latency mode, the state of the DQS signal is not taken into account by the OCTOSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in OCTOSPI_HLCR.

Figure 76. HyperBus write operation with no latency (register write)

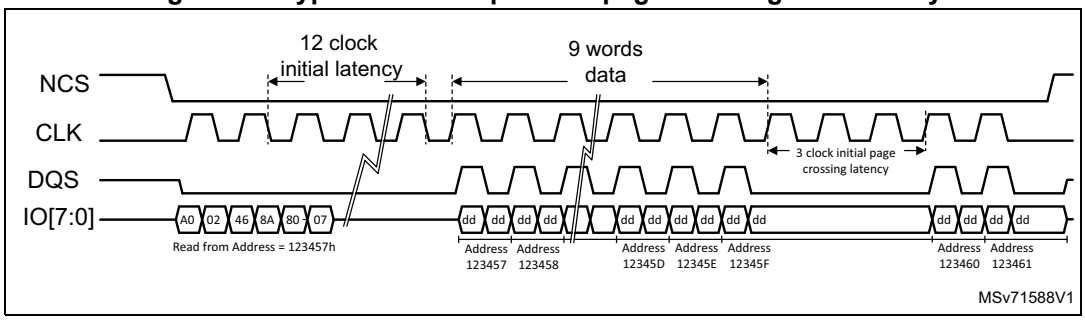


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 77. HyperBus read operation page crossing with latency



19.4.7 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The OCTOSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in OCTOSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated OCTOSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (OCTOSPI_CCR, OCTOSPI_TCR, OCTOSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the OCTOSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in OCTOSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Restarting after an interrupted transfer

When a read or write operation is interrupted by a timeout or communication regulation feature, the Octo-SPI interface, as soon as possible after getting back the port ownership, reissues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

19.4.8 OCTOSPI operating mode introduction

The OCTOSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

19.4.9 OCTOSPI indirect mode

In indirect mode, the commands are started by writing to the OCTOSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in OCTOSPI_CR, the OCTOSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading OCTOSPI_DR.

In indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The OCTOSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as

defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.

- If DL[31:0] = 0xFFFF FFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in OCTOSPI_SR, and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the OCTOSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (OCTOSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (OCTOSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (OCTOSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode passes through a 32-byte FIFO that is internal to the OCTOSPI. FLEVEL in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in OCTOSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in OCTOSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

19.4.10 OCTOSPI automatic status-polling mode

In automatic status-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in OCTOSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in OCTOSPI_CR, the AND-match mode is activated: SMF is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in OCTOSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the OCTOSPI is disabled (EN = 0).

OCTOSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in OCTOSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. The memory must then be configured in fixed latency.

19.4.11 OCTOSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the OCTOSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be

programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.

- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

19.4.12 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

19.4.13 OCTOSPI system configuration

The OCTOSPI is configured using OCTOSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.

The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and OCTOSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

19.4.14 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{\text{DEVSIZE+1}}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in OCTOSPI_HLCR.

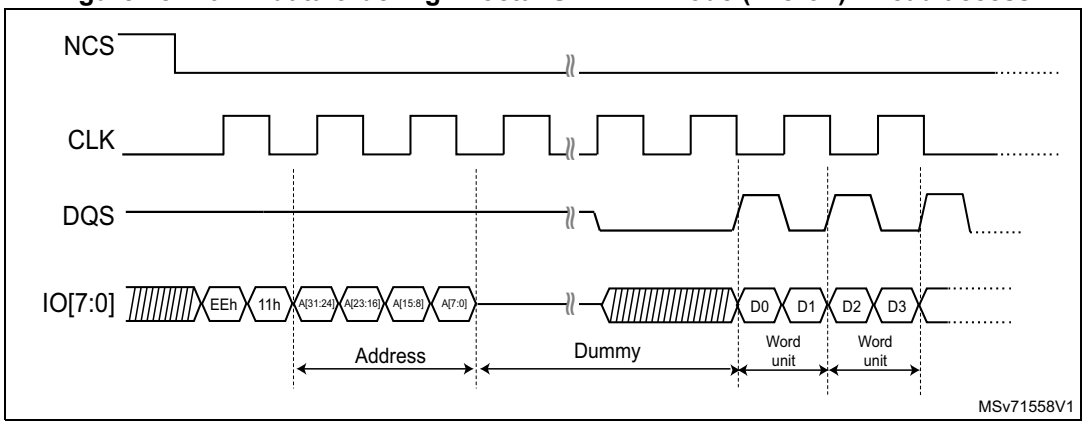
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the OCTOSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 78. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the

memory. Refer to *Figure 69: DTR read in octal-SPI mode with DQS (Macronix mode) example*.

- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 141: OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 79. OctaRAM read operation with reverse data ordering D1/D0

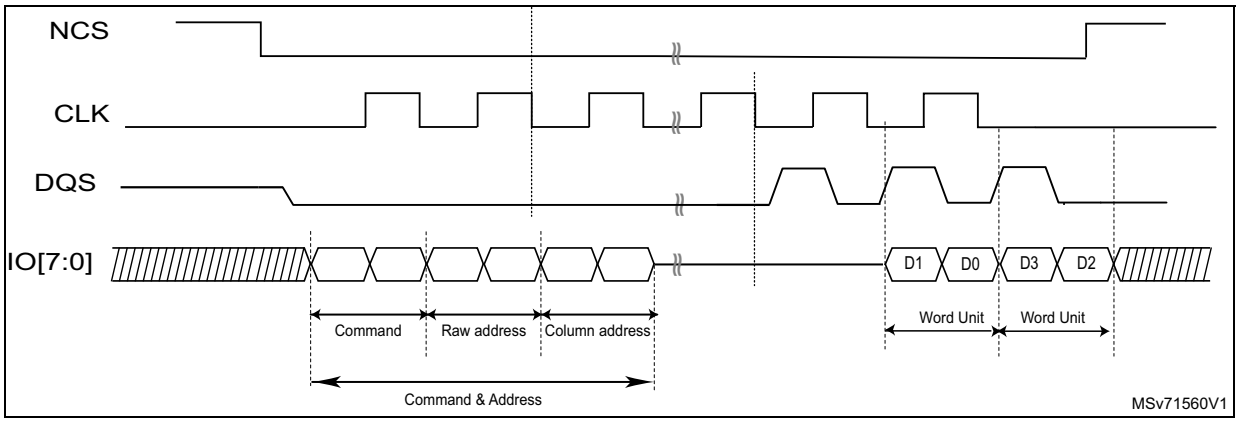


Table 141. OctaRAM command address bit assignment (based on 64-Mbyte(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64-Mbyte OctaRAM address assignment:
   Row Address [RA12:RA0]: 8K. Column address [CA9:CA0]: 1K. 64-Mbyte density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes

must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.

- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any others configurations described in this section. MTYP[2:0] = 010 for this standard mode.

19.4.15 OCTOSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000),
the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

When the OCTOSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.

7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

OCTOSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

OCTOSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay. This shift is performed by an external delay block located outside the OCTOSPI. The control of this feature depends on the device implementation (see the product reference manual for more details).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in OCTOSPI_DCR1.

Sending the instruction only once (SIOO)

A flash memory can provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. The user can take advantage of this type of features using SIOO in OCTOSPI_CCR.

SIOO is valid for memory-mapped mode only. If this bit is set, the instruction is sent only for the first command following a write to OCTOSPI_CCR.

Subsequent command sequences skip the instruction phase, until there is a write to OCTOSPI_CCR. SIOO has no effect when IMODE[1:0] = 00 (no instruction).

The SIOO mode is not supported when any of the communication regulations, NCS boundary, or refresh features are used.

19.4.16 OCTOSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the targeted address in OCTOSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

19.4.17 OCTOSPI error management

The following errors set the TEF flag in OCTOSPI_SR and generates an interrupt if enabled (TEIE = 1 in OCTOSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the OCTOSPI generates an AHB slave error in the following situations,:

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read or write address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing.
- The OCTOSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in OCTOSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode.

- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AHB bus transfer size (HSIZE) is different from 4 bytes (only for memory-mapped mode).
- Bit DMM is set in OCTOSPI_CR (dual-memory configuration) and octal mode is set.

19.4.18 OCTOSPI BUSY and ABORT

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in OCTOSPI_DR, it is recommended to disable the DMA channel before aborting the OCTOSPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

19.4.19 OCTOSPI reconfiguration or deactivation

Before any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

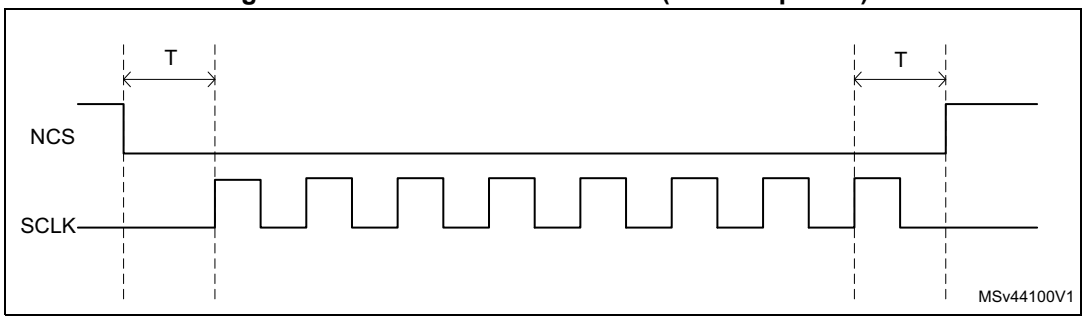
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

19.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

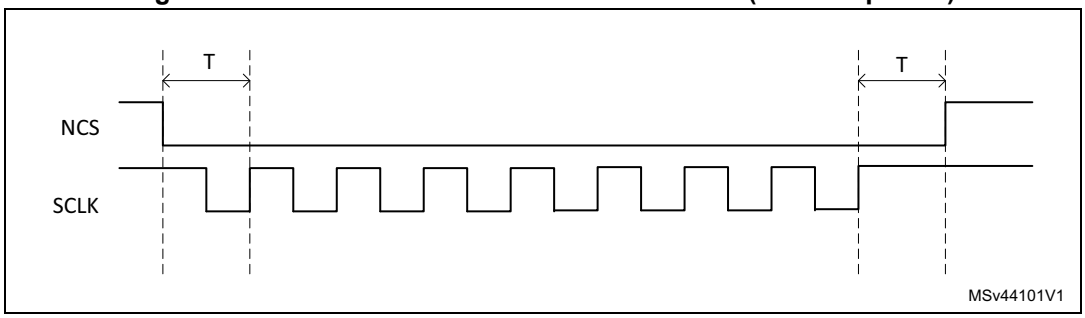
When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 80. NCS when CKMODE = 0 (T = CLK period)



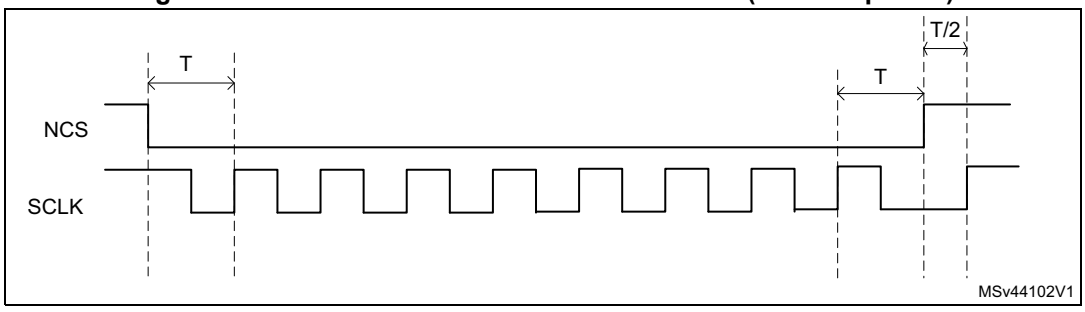
When CKMODE = 1 (clock mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 81. NCS when CKMODE = 1 in SDR mode (T = CLK period)



When the CKMODE = 1 (clock mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

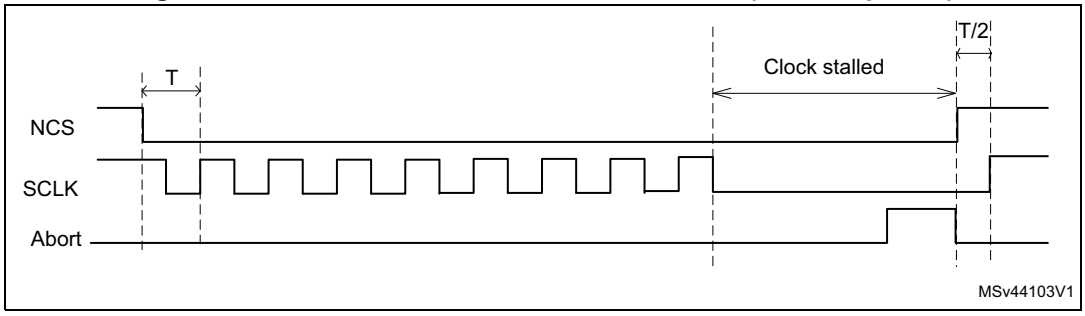
Figure 82. NCS when CKMODE = 1 in DTR mode (T = CLK period)



When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software services the FIFO. If an

abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later (see the figure below).

Figure 83. NCS when CKMODE = 1 with an abort (T = CLK period)



19.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 142. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.

Table 142. Address alignment cases (continued)

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

19.6 OCTOSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 143. OCTOSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

19.7 OCTOSPI registers

19.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FMODE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
		rw	rw					rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the OCTOSPI functional mode of operation.
00: Indirect-write mode
01: Indirect-read mode
10: Automatic status-polling mode (relevant in regular-command protocol only)
11: Memory-mapped mode
If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used to determine a match during the automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop

This bit determines if the automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the OCTOSPI.
1: Automatic status-polling mode stops as soon as there is a match.

Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable
This bit enables the status-match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level
This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.
00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.
00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.
...
11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.
Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 **MSEL**: External memory select
This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).
0: External memory 1 selected (data exchanged over IO[3:0])
1: External memory 2 selected (data exchanged over IO[7:4])
This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Bit 6 **DMM**: Dual-memory configuration
This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity
0: Dual-memory configuration disabled
1: Dual-memory configuration enabled
Note: This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the OCTOSPI.

0: OCTOSPI disabled

1: OCTOSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

19.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1)

Address offset: 0x0008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	MTYP[2:0]			Res.	Res.	Res.	DEVSIZE[4:0]				
					rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLY BYP	Res.	FRCK	CKMODE
		rw	rw	rw	rw	rw	rw					rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.

Others: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:

Number of bytes in device = $2^{[DEVSIZE+1]}$.

DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

In regular-command protocol, if DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).

1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.

0: CLK is not free running.

1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0/mode 3

This bit indicates the level taken by the CLK between commands (when NCS = 1).

0: CLK must stay low while NCS is high (chip-select released). This is referred to as clock mode 0.

1: CLK must stay high while NCS is high (chip-select released). This is referred to as clock mode 3.

19.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the OCTOSPI1_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

19.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

19.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFRESH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Others: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the divided clock period: if OCTOSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

19.7.6 OCTOSPI status register (OCTOSPI_SR)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag
This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

19.7.7 OCTOSPI flag clear register (OCTOSPI_FCR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the OCTOSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the OCTOSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the OCTOSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the OCTOSPI_SR register.

19.7.8 OCTOSPI data length register (OCTOSPI_DLR)

Address offset: 0x0040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect in memory-mapped mode.

19.7.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In HyperBus protocol, this field must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

19.7.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

19.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

19.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

Address offset: 0x0088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match
Value to be compared with the masked status register to get a match

19.7.13 OCTOSPI polling interval register (OCTOSPI_PIR)

Address offset: 0x0090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval
Number of CLK cycles between a read during the automatic status-polling phases

19.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

Address offset: 0x0100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIOO	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
rw		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **SIOO**: Send instruction only once mode
This bit has no effect when IMODE = 00 (see *Sending the instruction only once (SIOO)*).
0: Send instruction on every transaction
1: Send instruction only for the first command

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

19.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).
It is recommended to have at least six dummy cycles when using memories with DQS activated.

19.7.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

19.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address.

19.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period
After each access in memory-mapped mode, the OCTOSPI prefetches the subsequent bytes and hold them in the FIFO.
This bitfield indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

19.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Others: reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

19.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle

Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.

19.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

19.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

19.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

19.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31). It is recommended to have at least 5 dummy cycles when using memories with DQS activated.

19.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **INSTRUCTION[31:0]**: Instruction

Instruction to be sent to the external SPI device

19.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

19.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.
0: Latency on write accesses
1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.
0: Variable initial latency
1: Fixed latency

19.7.28 OCTOSPI register map

Table 144. OCTOSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	OCTOSPI_CR	Res.	Res.	FMDOE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
	Reset value			0	0					0	0		0	0	0	0	0				0	0	0	0	0	0	0			0	0	0	0
0x0004	Reserved	Reserved																															
0x0008	OCTOSPI_DCR1	Res.	Res.	Res.	Res.	Res.	MTYP [2:0]			Res.	Res.	Res.	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0				0	0	0	0	0			0	0	0	0	0	0					0		0	0

Table 144. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000C	OCTOSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x0010	OCTOSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x0014	OCTOSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018-0x001C	Reserved	Reserved																															
0x0020	OCTOSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x0024	OCTOSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x0028-0x003C	Reserved	Reserved																															
0x0040	OCTOSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	Reserved	Reserved																															
0x0048	OCTOSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	OCTOSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054-0x007C	Reserved	Reserved																															
0x0080	OCTOSPI_ PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	Reserved	Reserved																															
0x0088	OCTOSPI_ PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	Reserved	Reserved																															
0x0090	OCTOSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094-0x00FC	Reserved	Reserved																															

Table 144. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0100	OCTOSPI_CCR	SIOO	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0104	Reserved	Reserved																															
0x0108	OCTOSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x010C	Reserved	Reserved																															
0x0110	OCTOSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114-0x011C	Reserved	Reserved																															
0x0120	OCTOSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124-0x012C	Reserved	Reserved																															
0x0130	OCTOSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134-0x013C	Reserved	Reserved																															
0x0140	OCTOSPI_ WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0144	Reserved	Reserved																															
0x0148	OCTOSPI_ WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x014C	Reserved	Reserved																															
0x0150	OCTOSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0154-0x015C	Reserved	Reserved																															
0x0160	OCTOSPI_ WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0164-0x017C	Reserved	Reserved																															
0x0180	OCTOSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0

Table 144. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0184	Reserved	Reserved																															
0x0188	OCTOSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x018C	Reserved	Reserved																															
0x0190	OCTOSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0194-0x019C	Reserved	Reserved																															
0x01A0	OCTOSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01A4-0x01FC	Reserved	Reserved																															
0x0200	OCTOSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

20 Secure digital input/output MultiMediaCard interface (SDMMC)

20.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

20.2 SDMMC implementation

Table 145. SDMMC features

SDMMC modes/features(1)	SDMMC1
Variable delay (SDR104, HS200)	X
SDMMC_CKIN	X
SDMMC_CDIR, SDMMC_D0DIR	X
SDMMC_D123DIR	X

1. X = supported.

20.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 84. SDMMC "no response" and "no data" operations

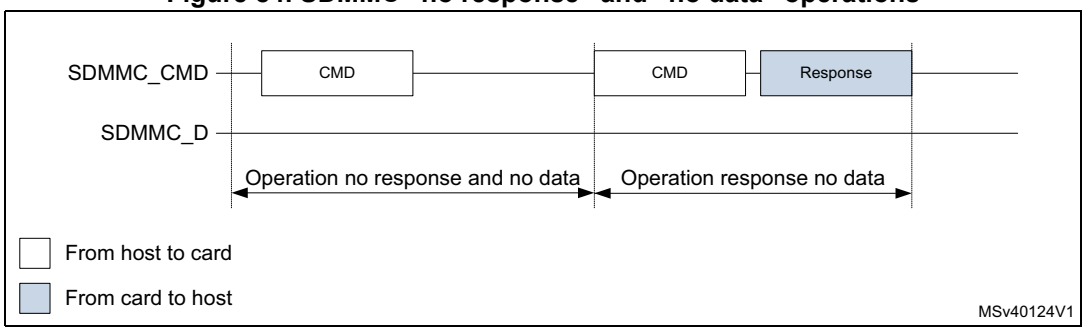
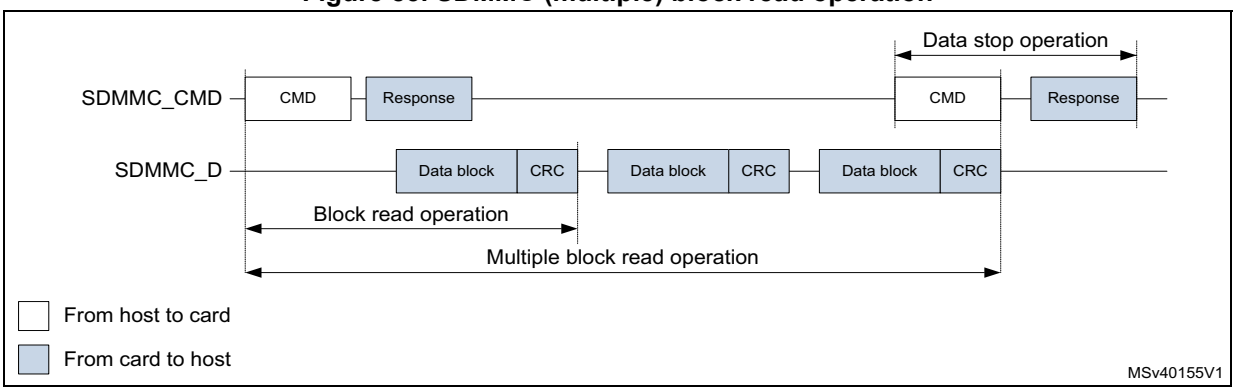
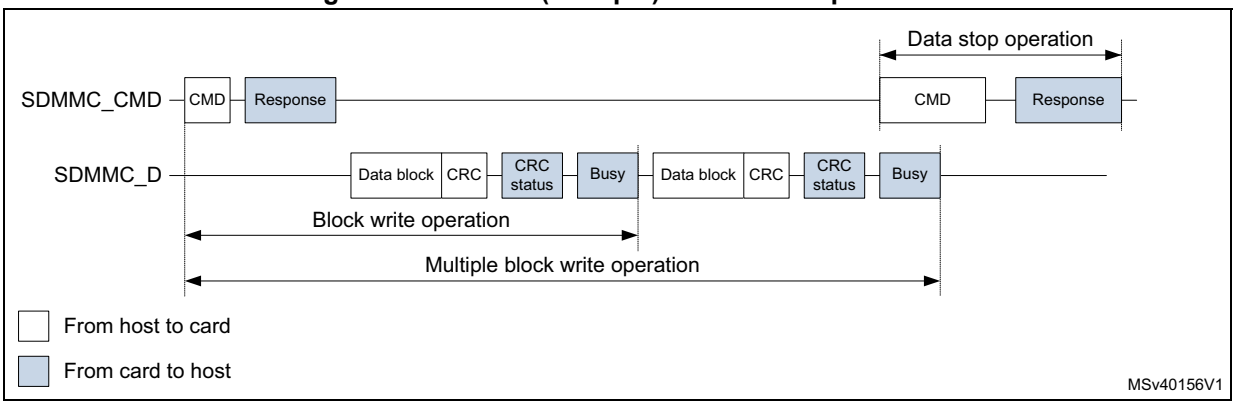


Figure 85. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 86. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 87. SDMMC (sequential) stream read operation

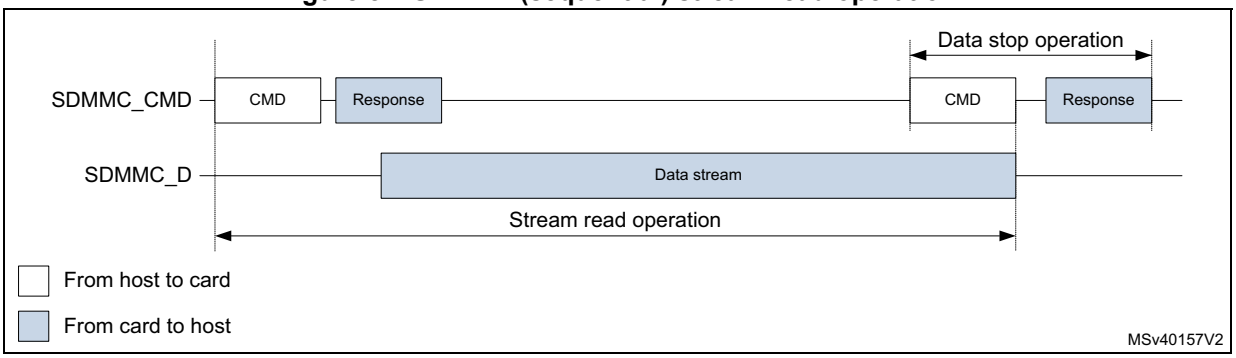
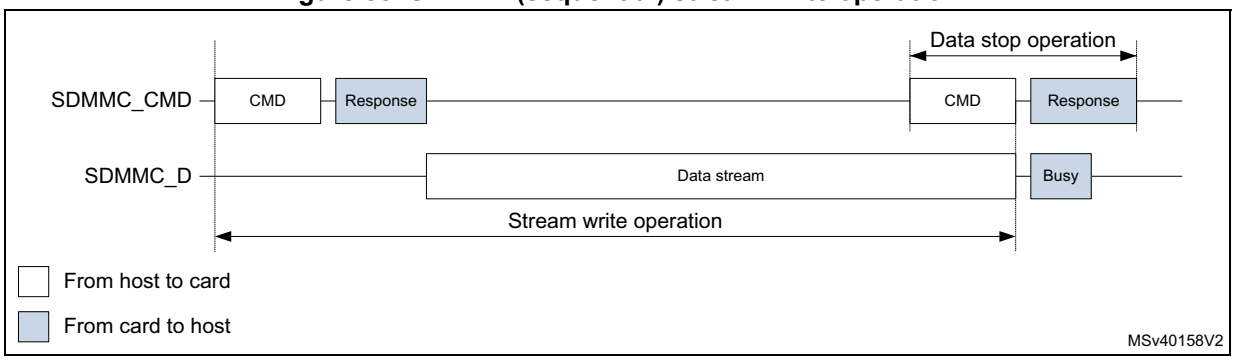


Figure 88. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

20.4 SDMMC operation modes

Table 146. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 147. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

20.5 SDMMC functional description

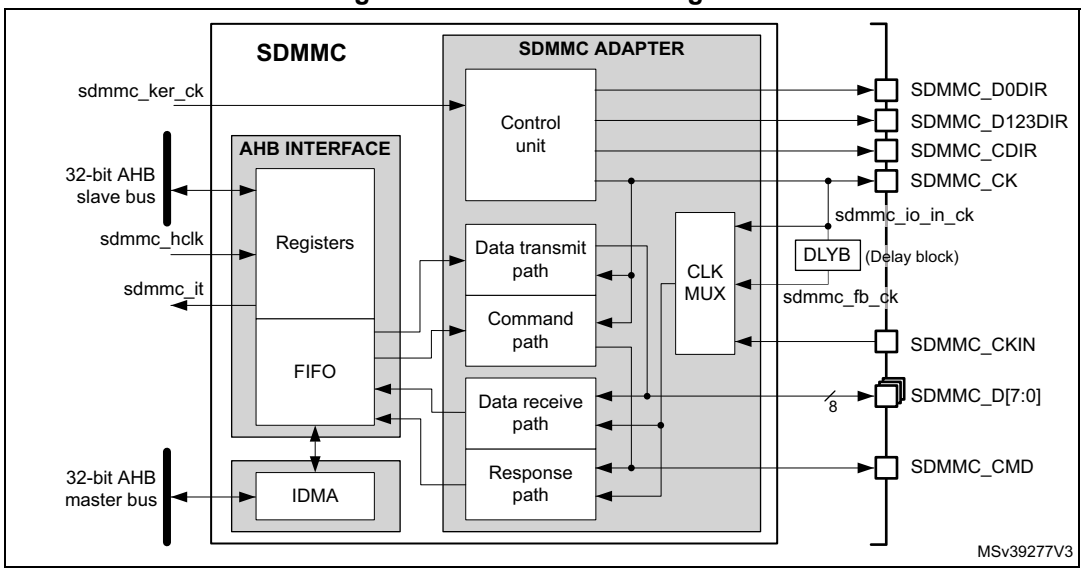
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

20.5.1 SDMMC block diagram

Figure 89 shows the SDMMC block diagram.

Figure 89. SDMMC block diagram



20.5.2 SDMMC pins and internal signals

Table 148 lists the SDMMC internal input/output signals, *Table 149* the SDMMC pins (alternate functions).

Table 148. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 148. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 149. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

20.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 90*.

Figure 90. SDMMC Command and data phase relation

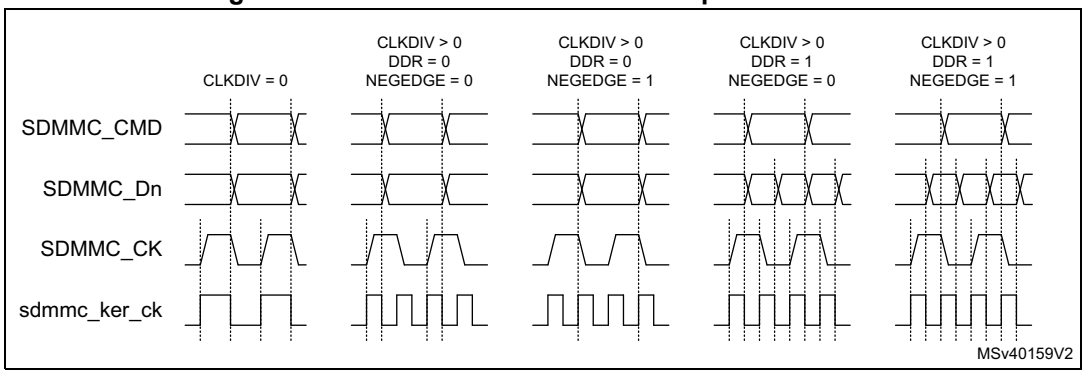


Table 150. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	Generated on sdmmc_ker_ck falling edge	
>0	0	0	Generated on sdmmc_ker_ck rising edge	Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

20.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 89: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

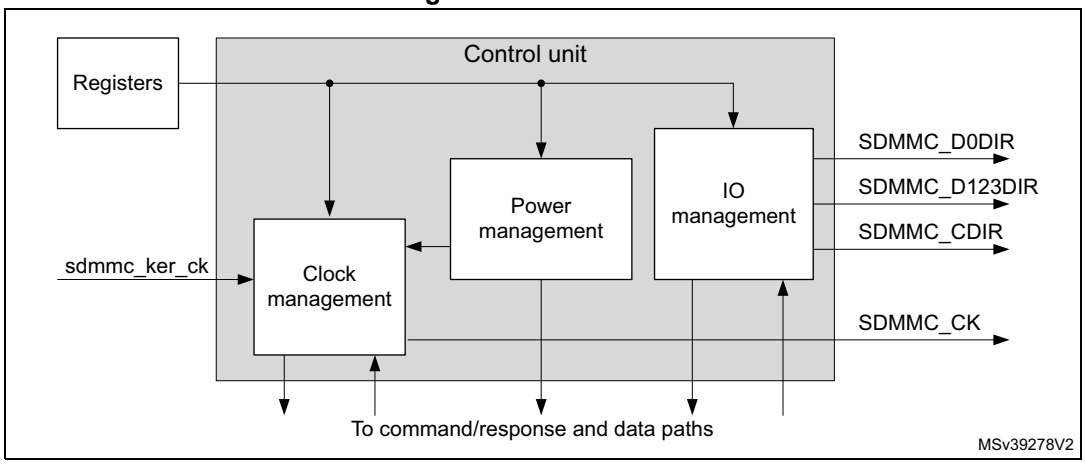
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 91*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 91. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

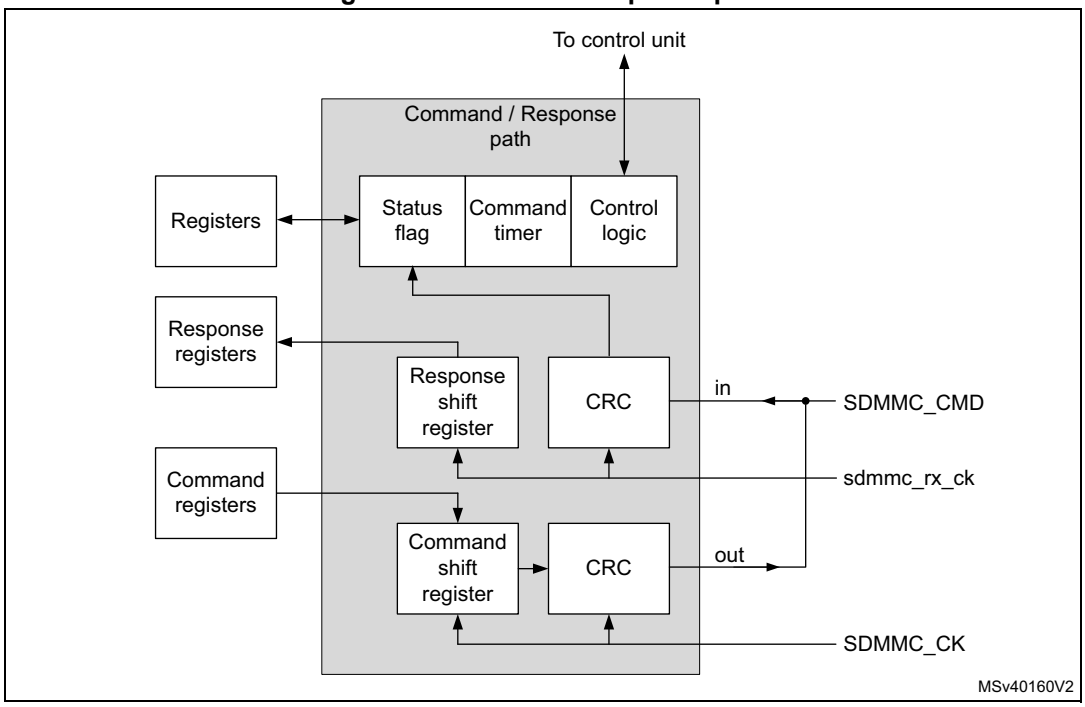
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 92. Command/response path

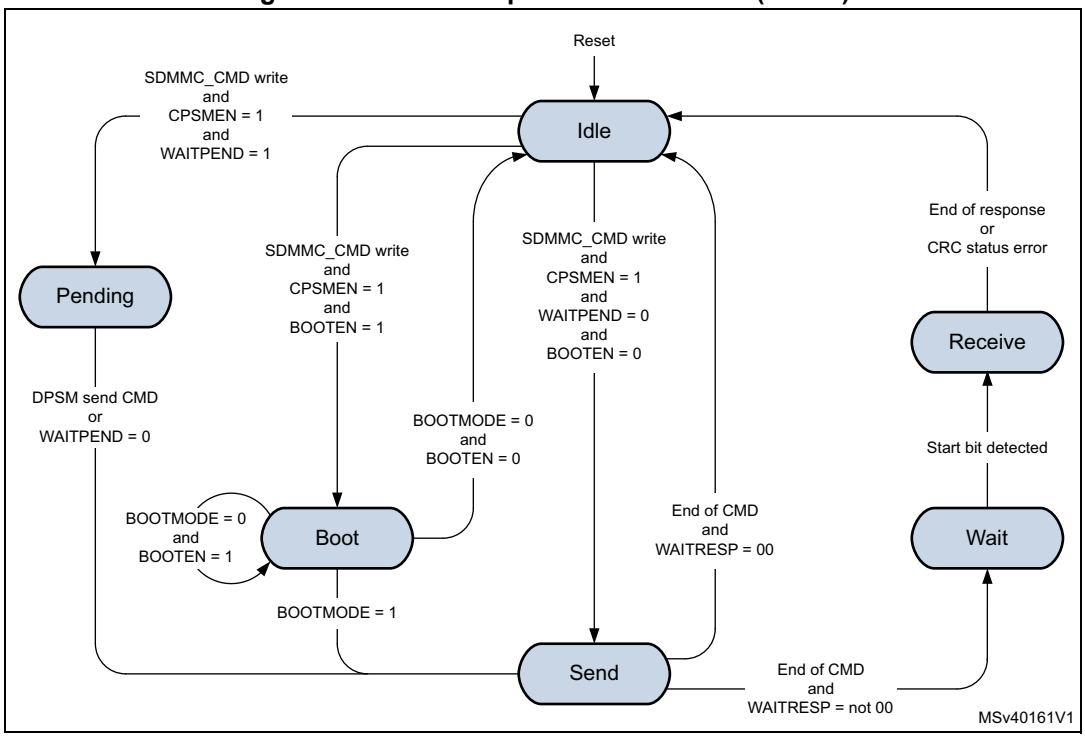


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 93. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 151.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 151. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 152*, *Table 153* and *Table 154*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 152. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 153. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 154. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 90*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder $[(M(x) * x^7) / G(x)]$

$G(x) = x^7 + x^3 + 1$

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + \ldots + (\text{last bit before CRC}) * x^0$

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 155*.

Table 155. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 156*:

Table 156. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 157*:

Table 157. Command path error handling

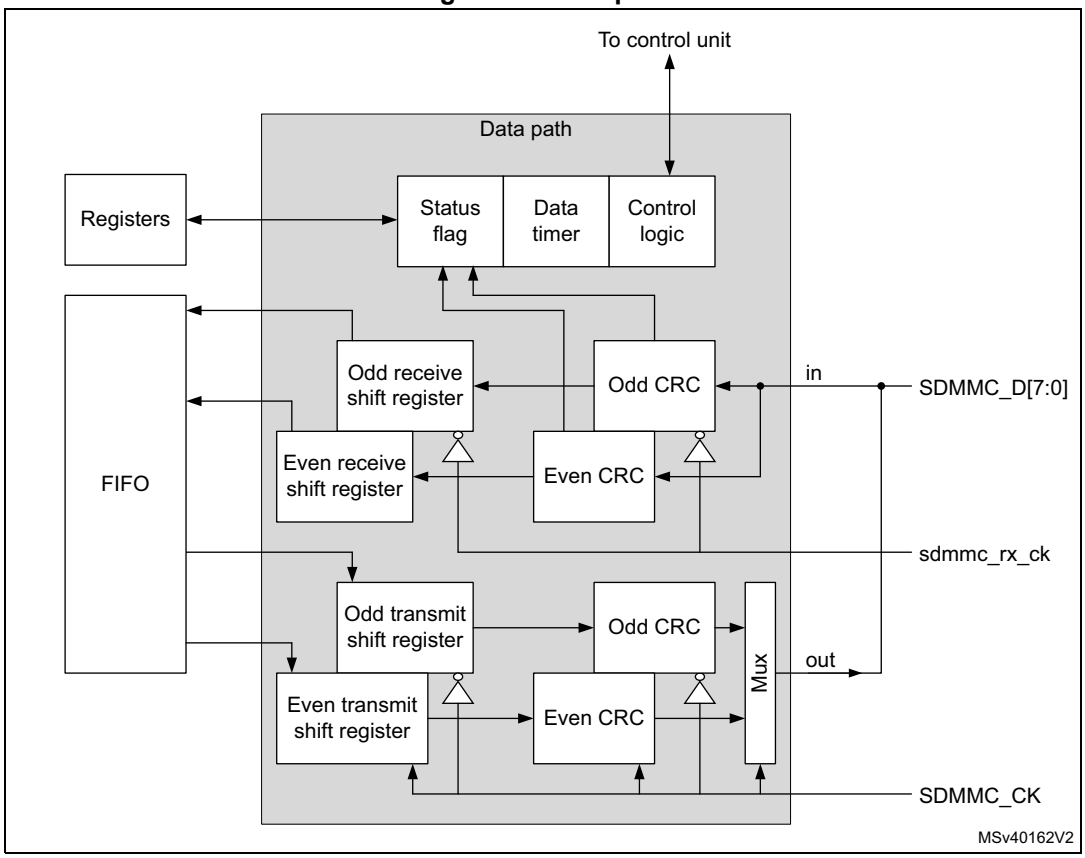
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 94* shows the data path block diagram.

Figure 94. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 95* and *Figure 96*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 95. DDR mode data packet clocking

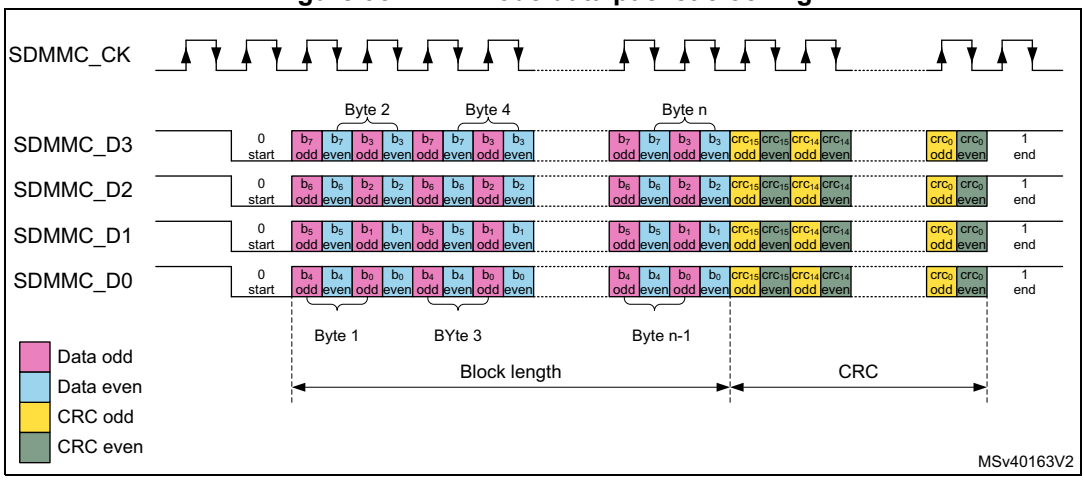
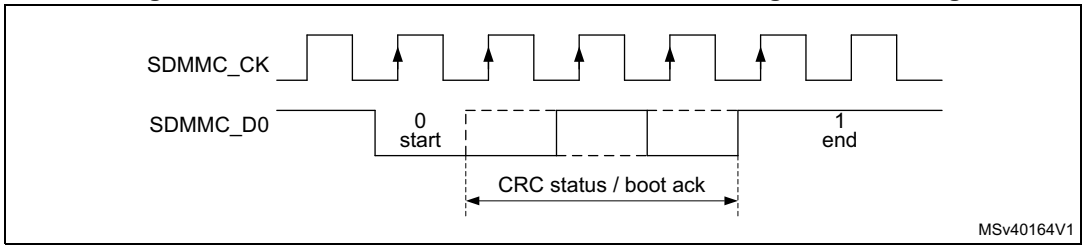


Figure 96. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

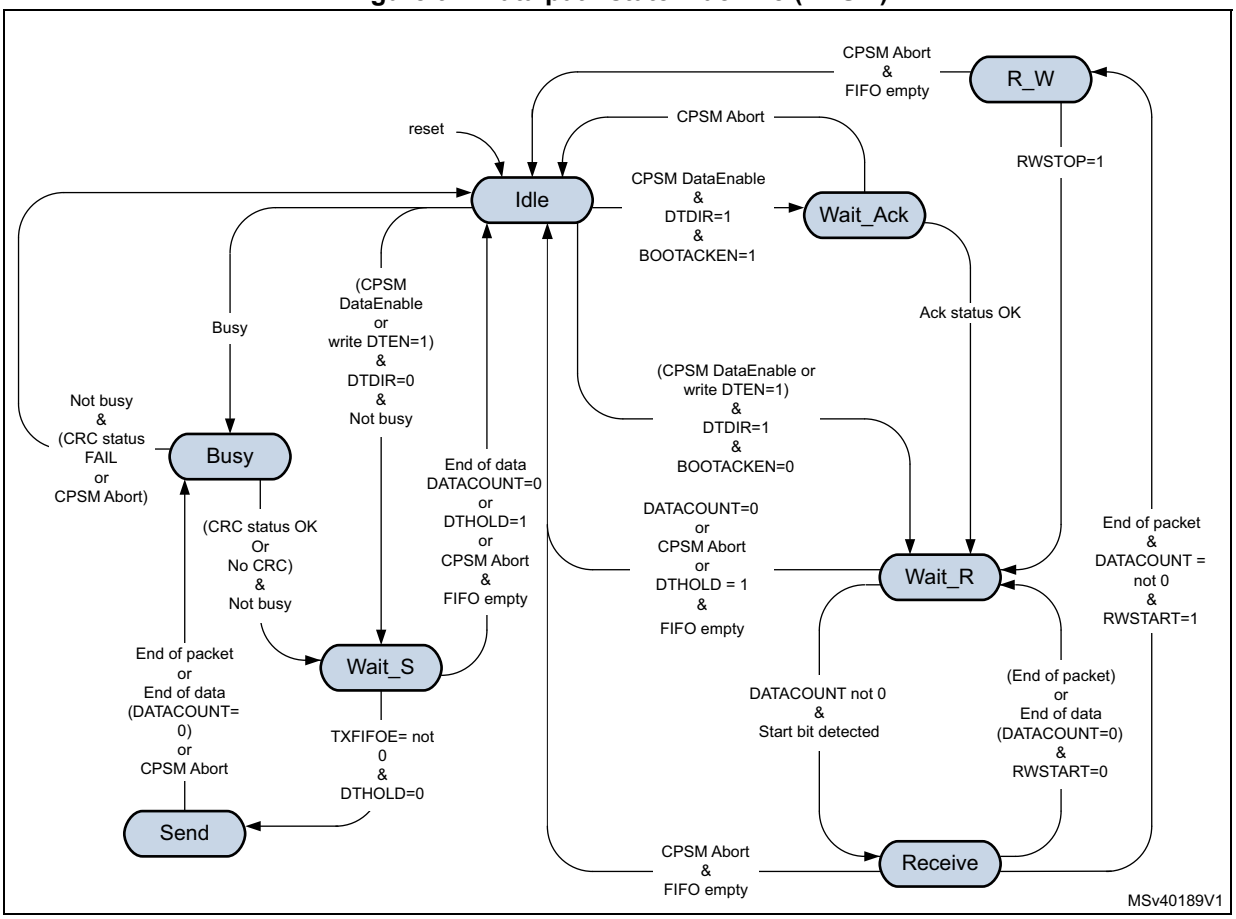
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 97*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 97. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.

- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.

– When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  – The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  – If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.

  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise.

  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.

  – In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  – When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  – When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  – When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 158. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 159*:

Table 159. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 159. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 160*:

Table 160. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 161*.

Table 161. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
3. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   – The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.

- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 162*.

Table 162. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 163*.

Table 163. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

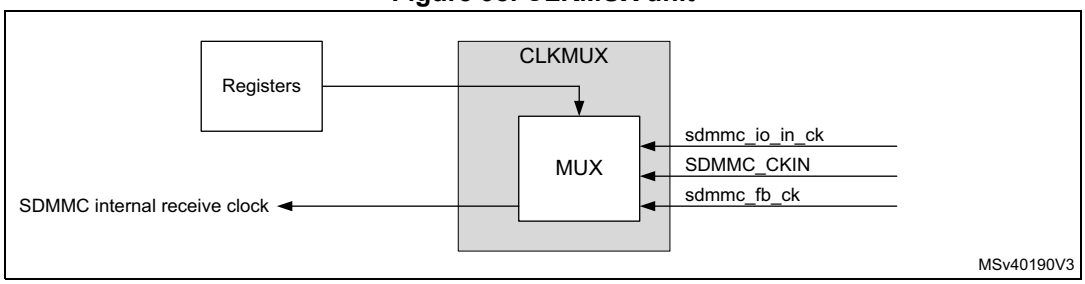
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 98. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

20.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.
- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

20.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer.

When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

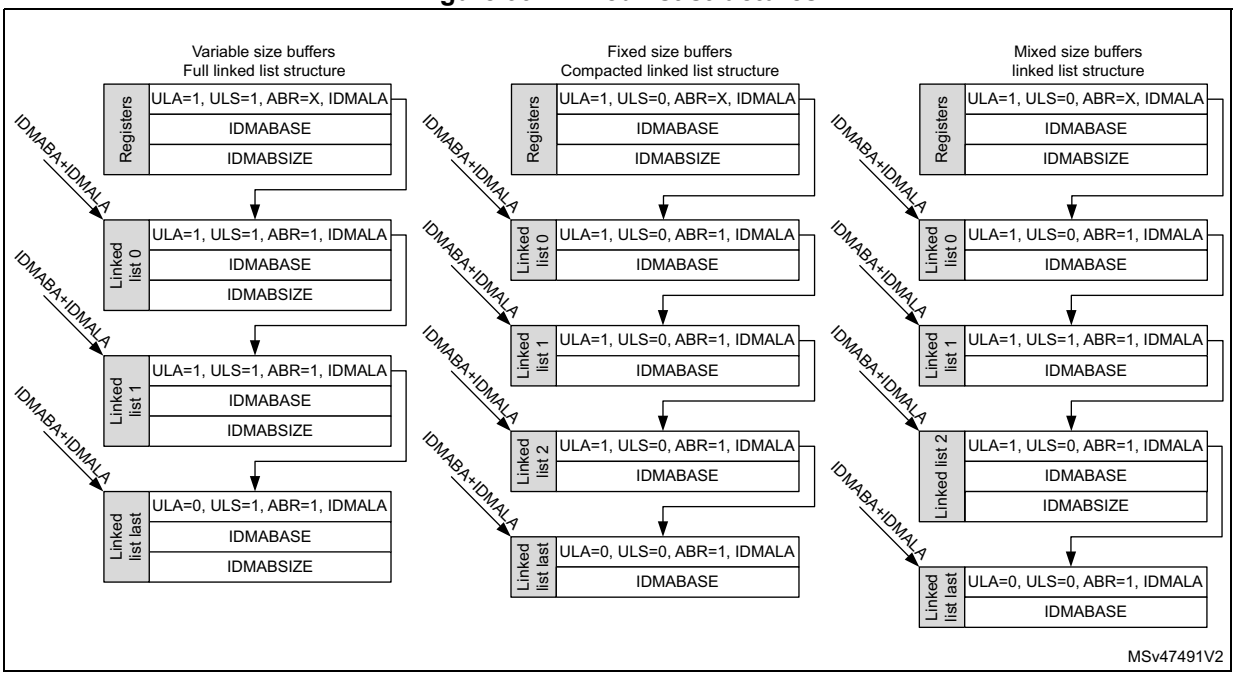
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 99. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

20.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth: for example, for SDR50 4-bit mode (50 Mbyte/s), the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

Table 164. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	150
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5
SD SDR104	4	208	78

20.6 Card functional description

20.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 165. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

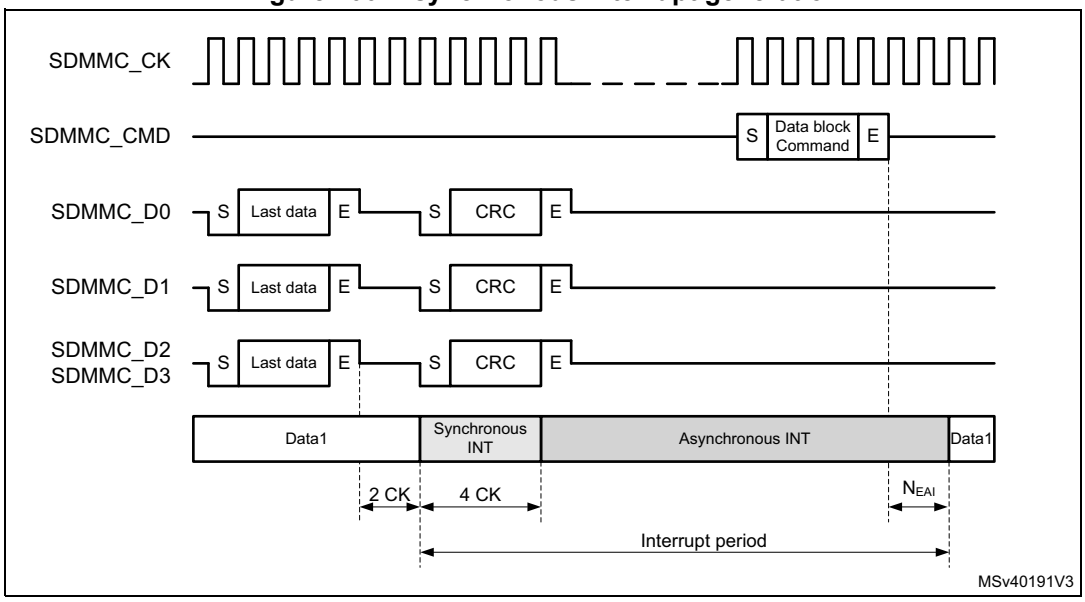
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 165*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 100. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 101. Synchronous interrupt period data read

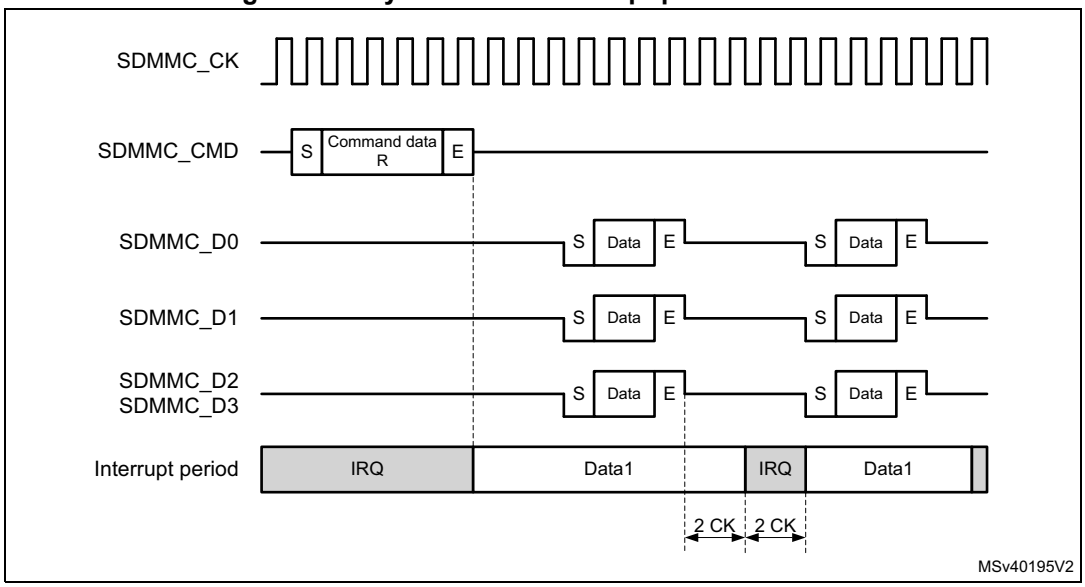
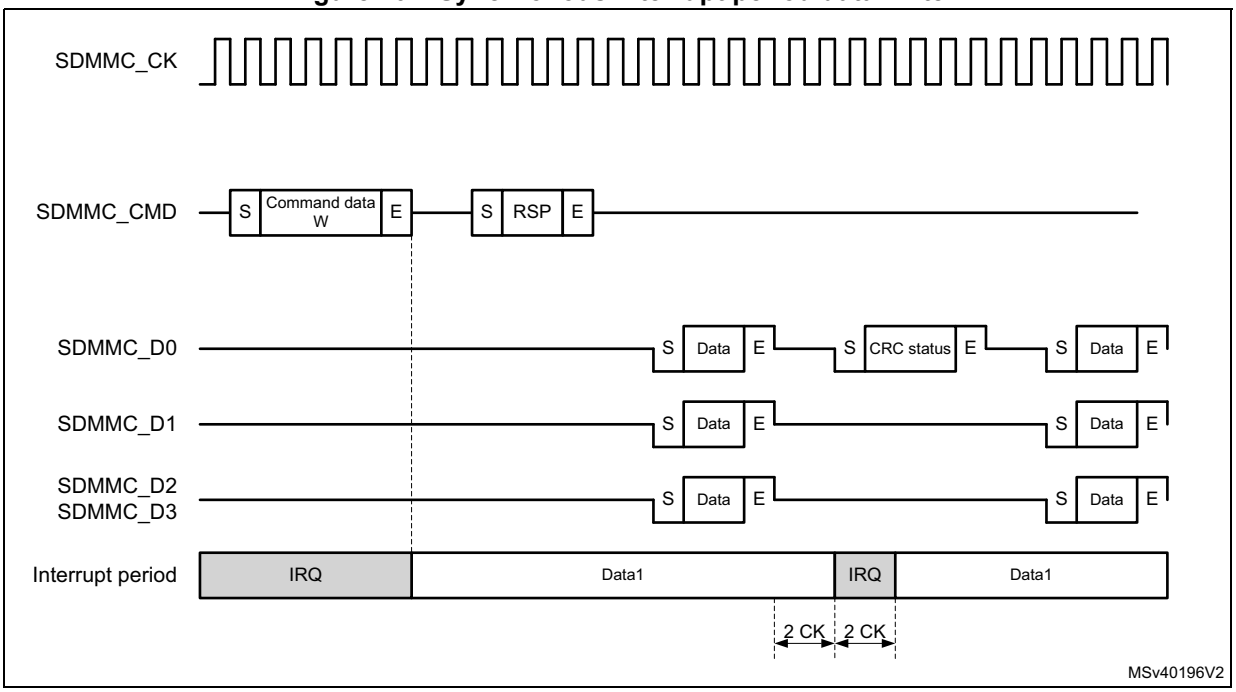


Figure 102. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 103. Asynchronous interrupt period data read

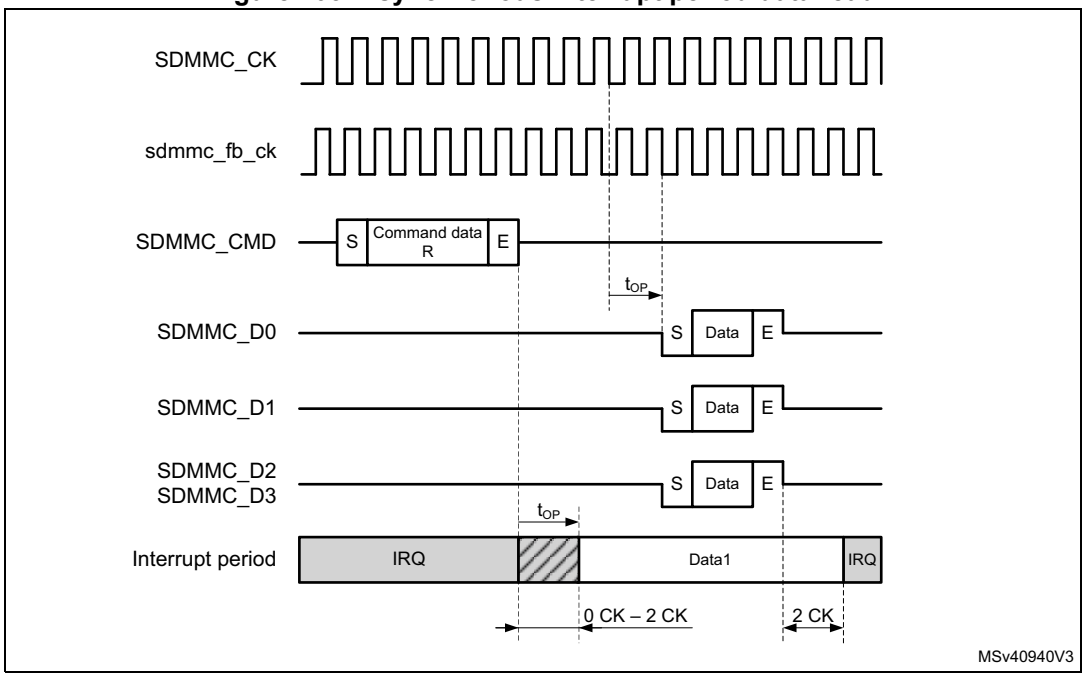
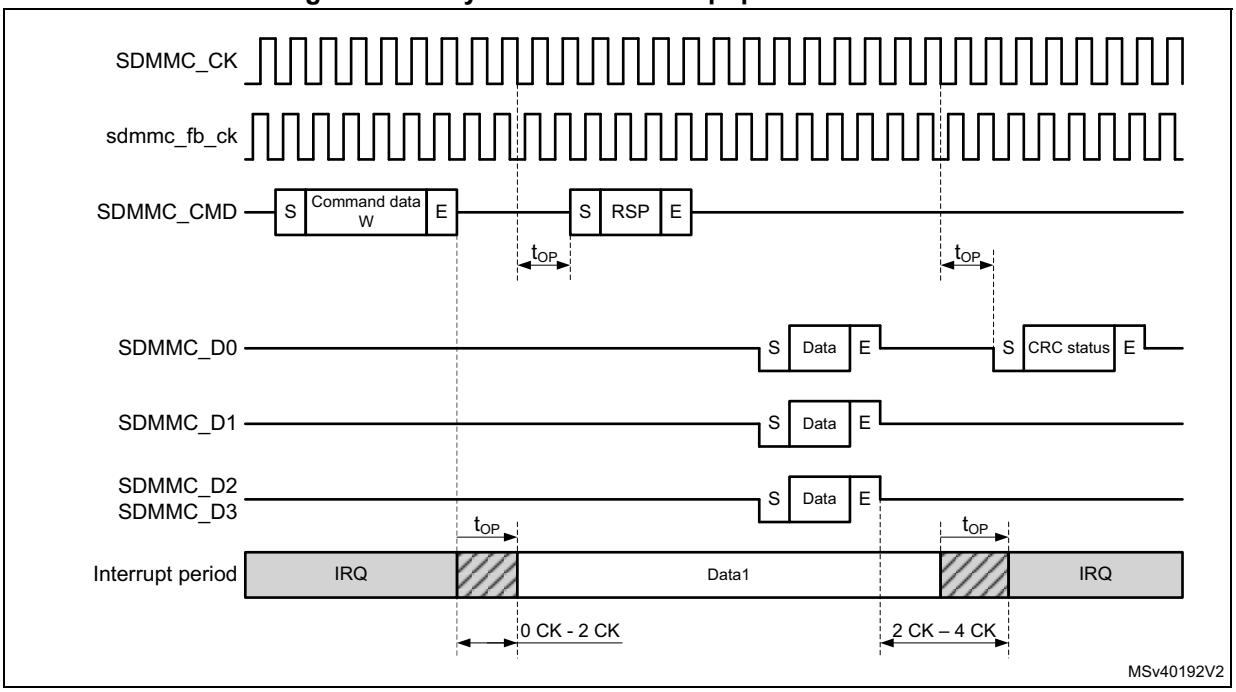


Figure 104. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 166*.

Table 166. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

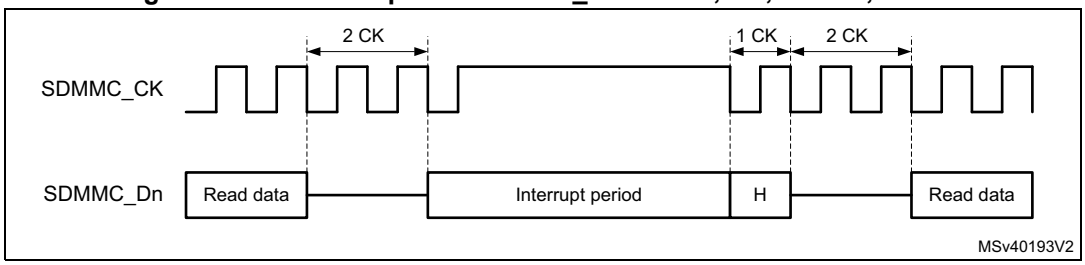
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 165*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

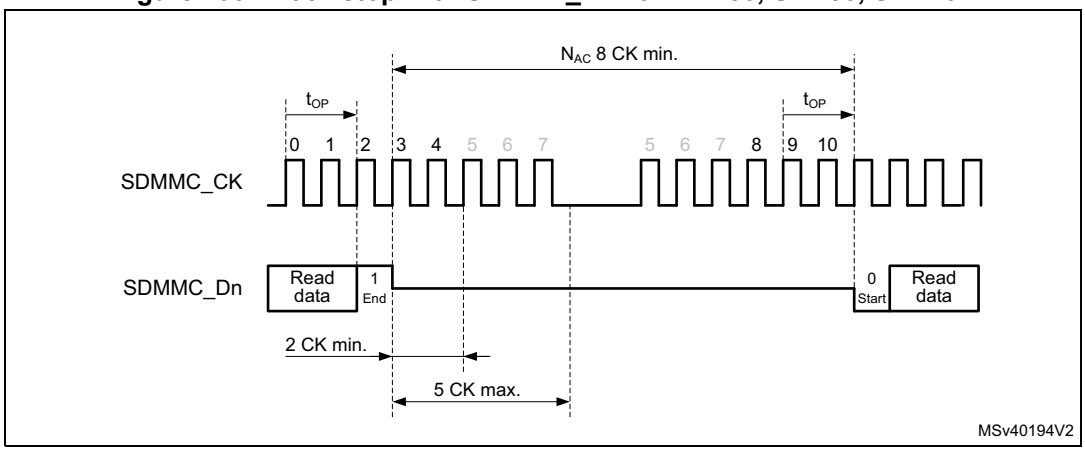
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 105*).

Figure 105. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 106*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 106. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



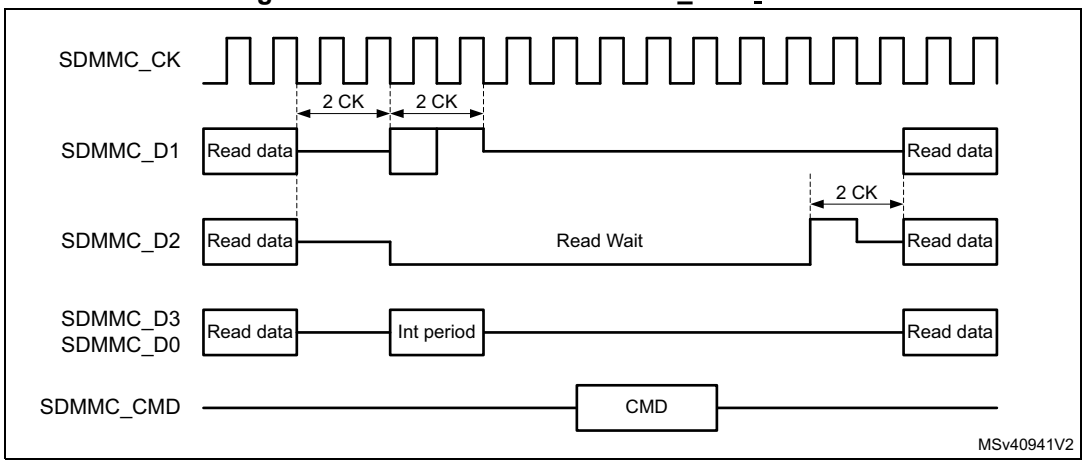
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

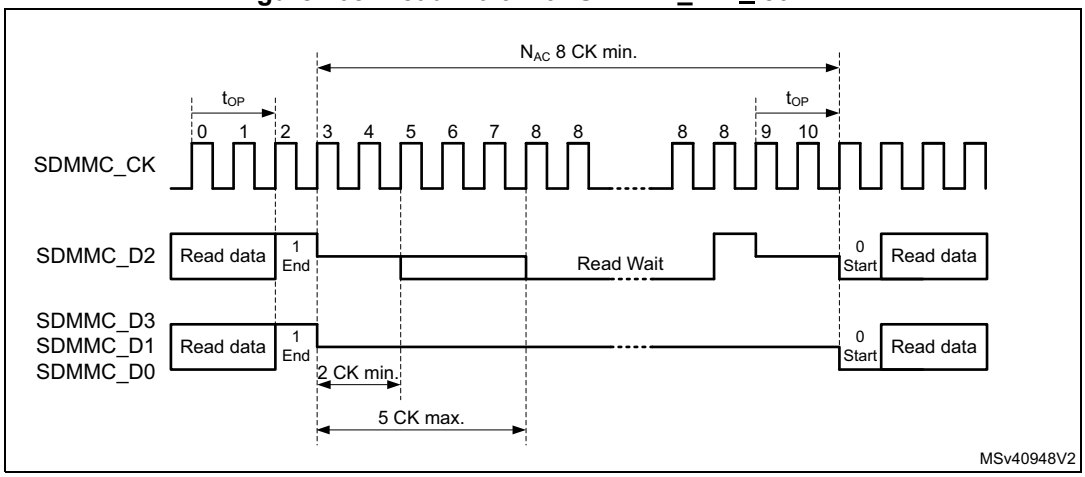
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 107*.

Figure 107. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 108*).

Figure 108. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

20.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 167. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

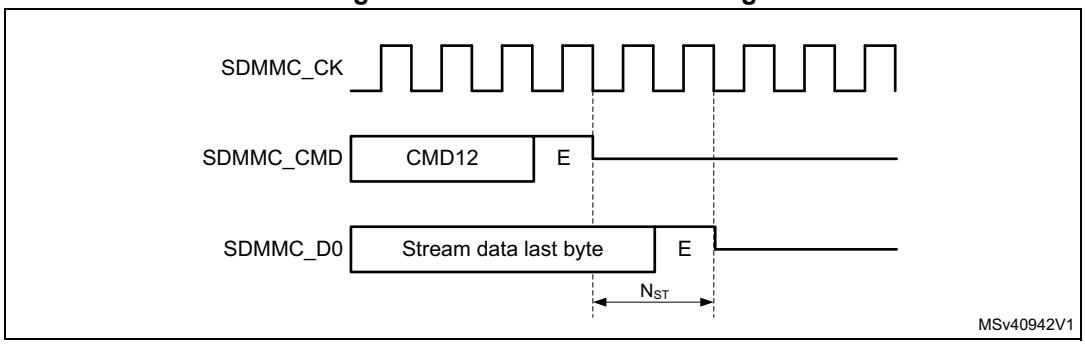
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 109. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

20.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

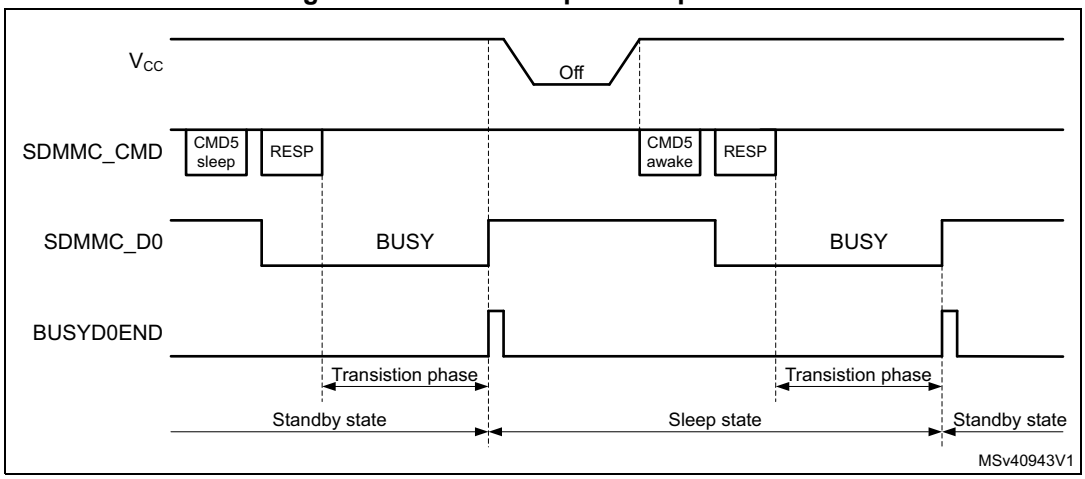
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 110. CMD5 Sleep Awake procedure



20.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

20.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

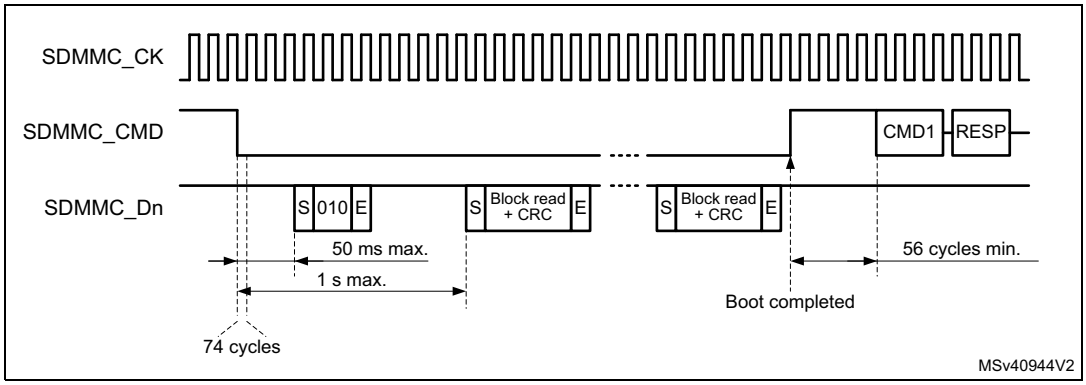
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 111. Normal boot mode operation



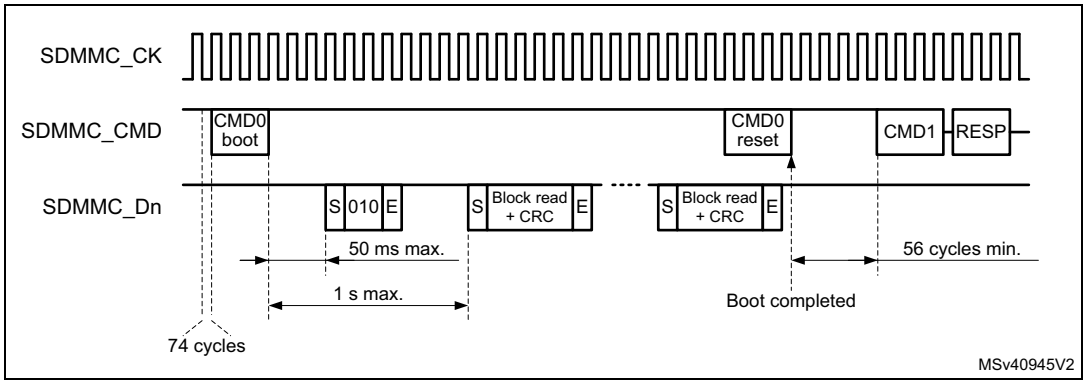
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   – the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   – the ACK timeout to start.
   – DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   – when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   – when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   – when data CRC fails the DCRCFAIL flag is also generated.
   – when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   – If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 112. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

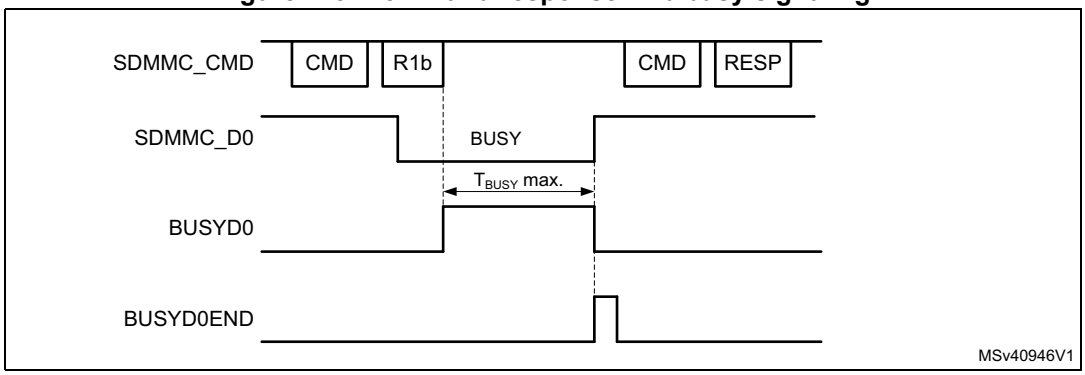
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

20.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 113. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

20.6.7 Reset and card cycle power

Reset

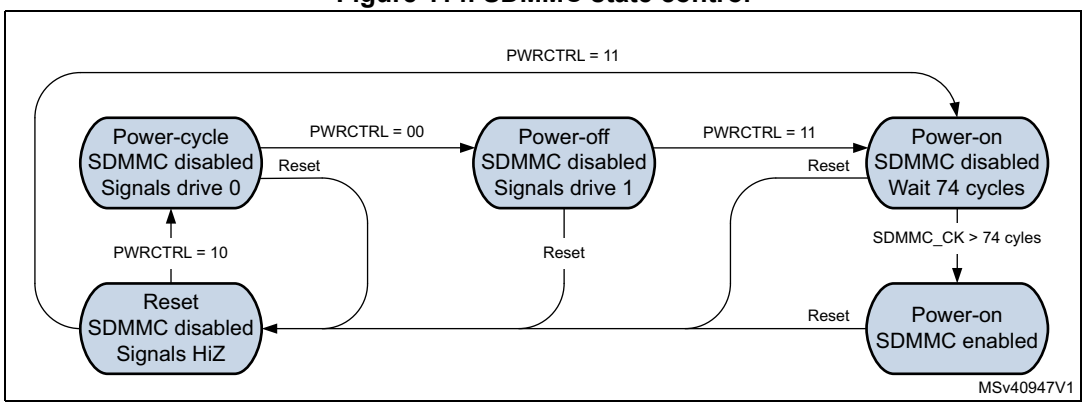
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 114.*.

Figure 114. SDMMC state control

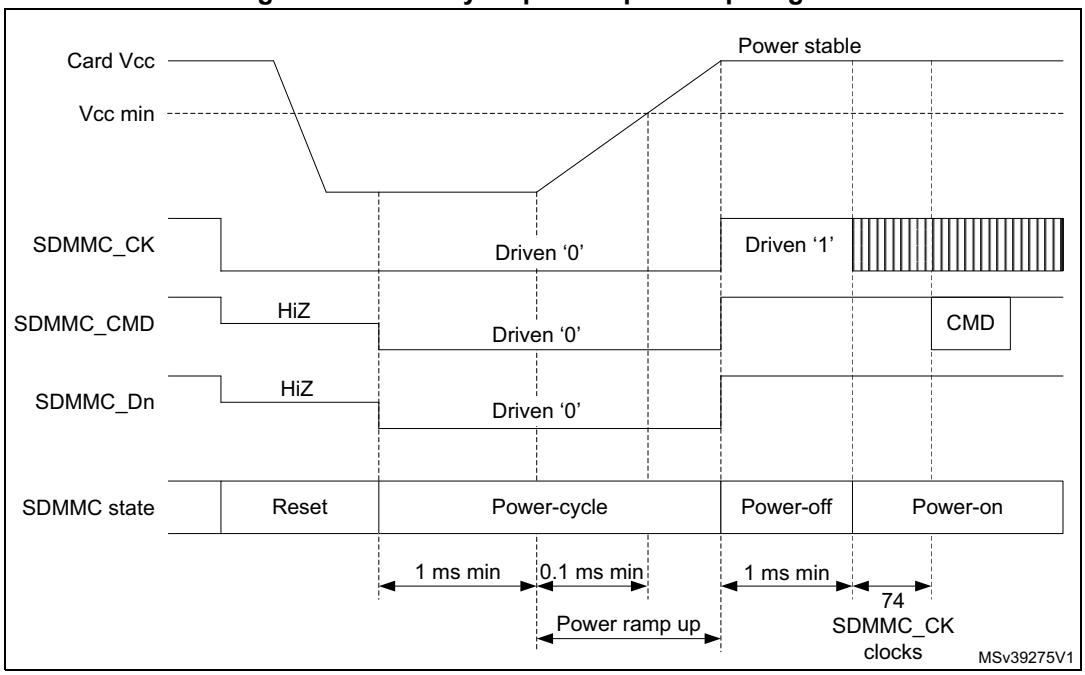


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 115. Card cycle power / power up diagram



20.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

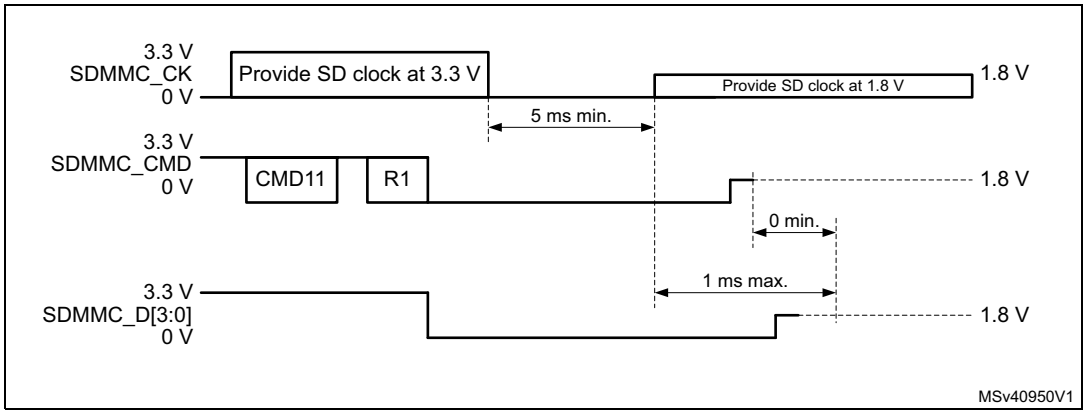
Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can not be used.

20.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch

sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 116. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   - When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

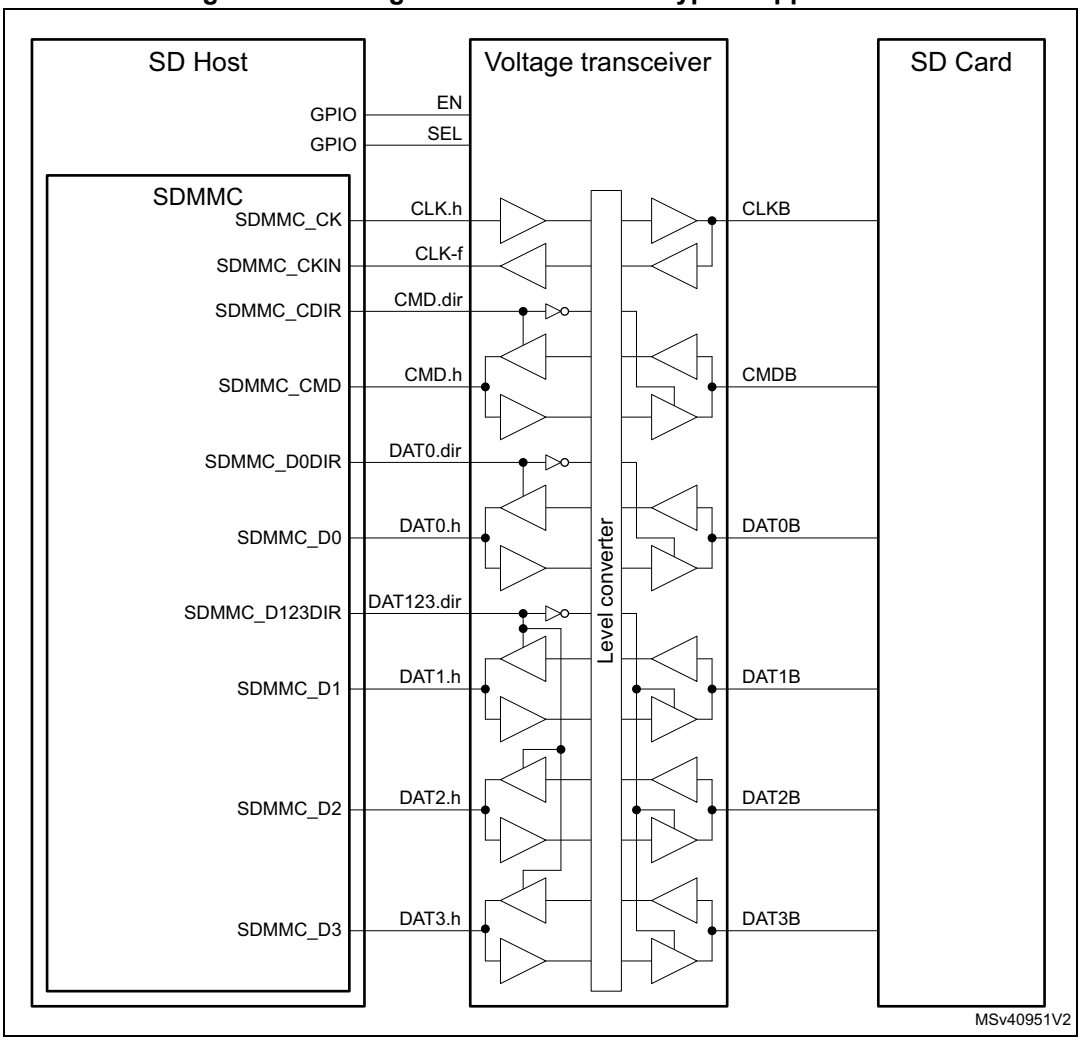
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 117. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

20.9 SDMMC interrupts

Table 168. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

20.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

20.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR POL	VSWI TCHEN	VSWI TCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

20.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 20.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

20.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

20.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

20.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

20.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTATUS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTATUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 169*.

The card status size is 32 or 128 bits, depending on the response type.

Table 169. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

20.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATATIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

20.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATALENGTH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

20.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Define the data block length when the block data transfer mode is selected:

0000: Block length = 2^0 = 1 byte

0001: Block length = 2^1 = 2 bytes

0010: Block length = 2^2 = 4 bytes

0011: Block length = 2^3 = 8 bytes

0100: Block length = 2^4 = 16 bytes

0101: Block length = 2^5 = 32 bytes

0110: Block length = 2^6 = 64 bytes

0111: Block length = 2^7 = 128 bytes

1000: Block length = 2^8 = 256 bytes

1001: Block length = 2^9 = 512 bytes

1010: Block length = 2^{10} = 1024 bytes

1011: Block length = 2^{11} = 2048 bytes

1100: Block length = 2^{12} = 4096 bytes

1101: Block length = 2^{13} = 8192 bytes

1110: Block length = 2^{14} = 16384 bytes

1111: Reserved

When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)

When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

00: Block data transfer ending on block count.

01: SDIO multibyte data transfer.

10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)

11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: From host to card.

1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.

This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).

0: Do not start data transfer without CPSM data transfer command.

1: Start data transfer without CPSM data transfer command.

20.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and

when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

20.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

20.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

20.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

20.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACKTIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

20.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMAB MODE	IDMA EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IDMABMODE**: Buffer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

20.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABNDT[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:5 **IDMABNDT[11:0]**: Number of bytes per buffer
This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

20.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABASE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

20.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMALA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

20.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABA[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

20.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

20.10.21 SDMMC register map

Table 170. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																

Table 170. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_ RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_ RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_ DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_ DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SDMMC_ DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_ STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0

Table 170. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11:0]												Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0					
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																													Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 97* for the register boundary addresses.

21 Delay block (DLYB)

21.1 Introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Octo-SPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

21.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

21.3 DLYB implementation

Table 171. STM32U3 series features

DLYB associated peripheral	DLYBOS1	DLYBSD1
OCTOSPI1	X	-
SDMMC1	-	X

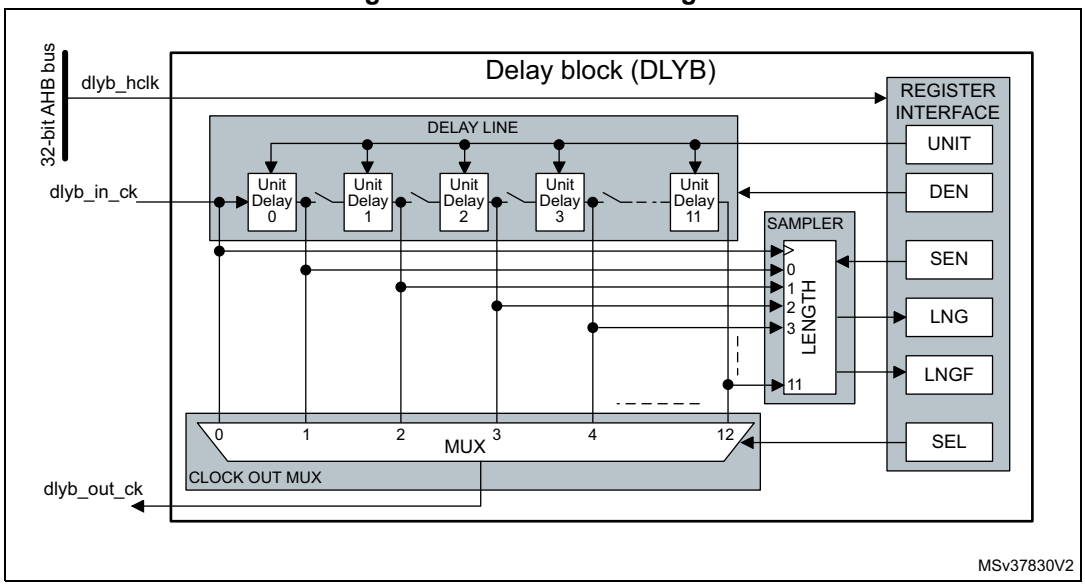
21.4 DLYB functional description

21.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 118. DLYB block diagram



21.4.2 DLYB pins and internal signals

Table 172 lists the DLYB internal signals.

Table 172. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

21.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 173. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

21.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

21.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

Octo-SPI use case:

The delay block is used in conjunction with Octo-SPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small "Unit" gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

SDMMC use case:

The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

21.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

21.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

21.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.

These bits can only be written when SEN = 1.

Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.

These bits can only be written when SEN = 1.

Output clock phase = input clock + SEL[3:0] x unit delay

21.5.3 DLYB register map

Table 174. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

22 Analog-to-digital converters (ADC)

22.1 Introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master)

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 19 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

22.2 ADC main features

- High-performance features
  - Up to 2 ADCs which can operate in dual mode
    ADC1 is connected to 15 external channels + 4 internal channels
    ADC2 is connected to 13 external channels + 6 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - AHB slave bus interface to allow fast data handling
  - Offset calibration support
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Data can be routed to ADF for post processing
  - Four dedicated data registers for the injected channels

- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 15 GPIO pads
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel for monitoring V_{CORE} internal voltage
  - Connection to DAC internal channels
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 119 shows the block diagram of one ADC.

22.3 ADC implementation

Table 175. ADC features

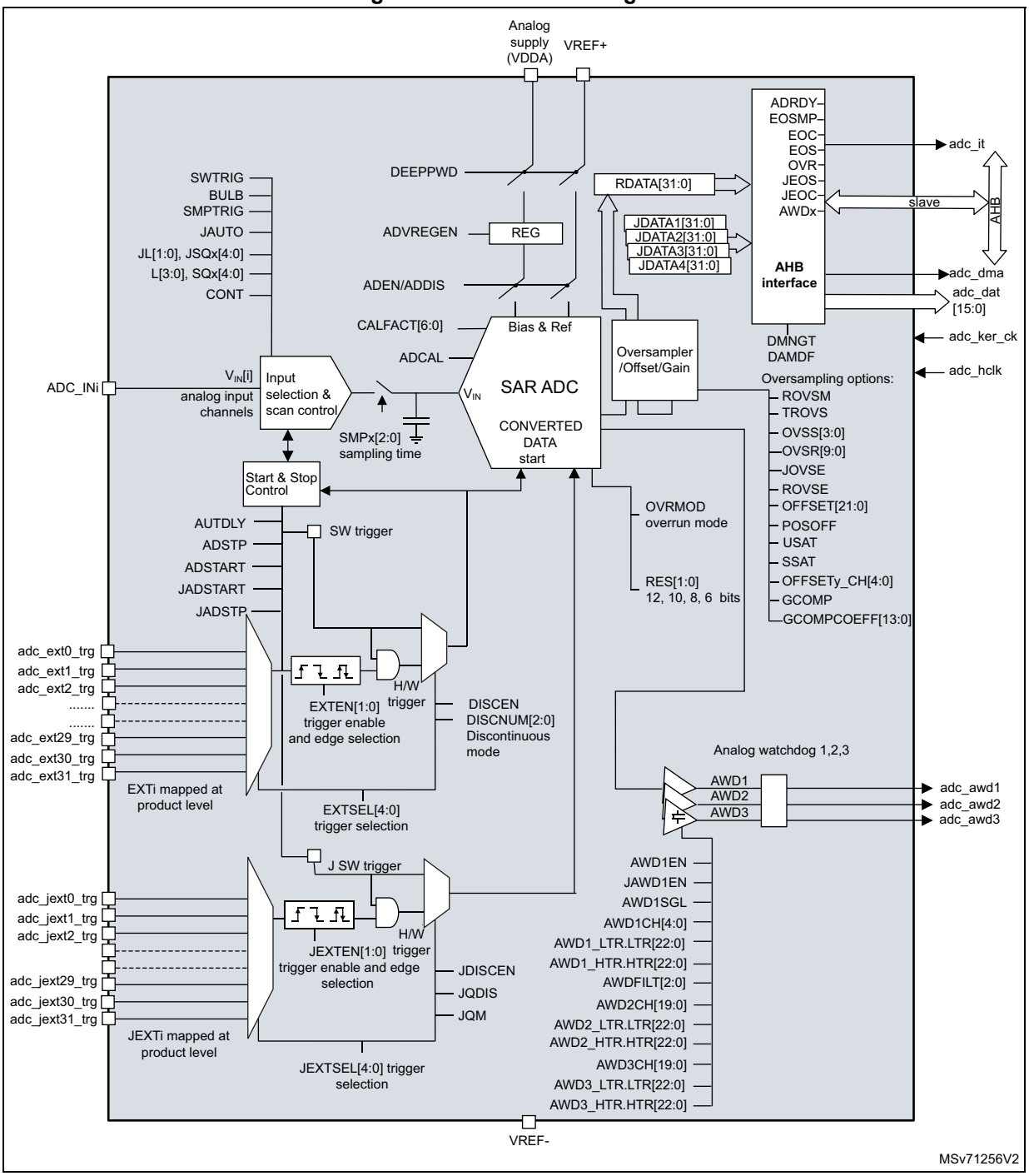
ADC modes/features	ADC1, ADC2
Resolution	12 bits
Maximum sampling speed	2 Msps (12-bit resolution)
Dual mode operation	X
Offset calibration	X
Single-end input	X
Differential input	-
Injected channel conversion	X
Oversampling	up to x1024
Data register	32 bits
DMA support	X
Parallel data output to ADF	X
Offset compensation	X
Gain compensation	X
Number of analog watchdog	3

22.4 ADC functional description

22.4.1 ADC block diagram

Figure 119 shows the ADC block diagram and *Table 176* gives the ADC pin description.

Figure 119. ADC block diagram



22.4.2 ADC pins and internal signals

Table 176. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
ADCx_INi	External analog input signals	19 external analog input channels (refer to *Section 22.4.4: ADC connectivity* for details)

Table 177. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{IN}i$	Analog input channels	Internal analog input channels connected either to ADCx_IN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3)
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
adc_dat[15:0]	Output	ADC data outputs

Table 178. ADC1/2 interconnection

Signal name	Source/destination
ADC1 VIN[0], ADC2 VIN[0]	VREFINT (output voltage from internal reference voltage)
ADC1 VIN[16], ADC2 VIN[16]	VBAT/4 (VBAT pin input voltage divided by 4)
ADC1 VIN[17], ADC2 VIN[17]	VSENSE (internal temperature sensor output voltage)
ADC1 VIN[18], ADC2 VIN[18]	VCORE (internal logic core voltage)
ADC2 VIN[14]	dac1_out1 (DAC internal channel 1)
ADC2 VIN[15]	dac1_out2 (DAC internal channel 2)

Table 178. ADC1/2 interconnection (continued)

Signal name	Source/destination
adc_dat1[15:0]	adf_adc1_dat[15:0]
adc_dat2[15:0]	adf_adc2_dat[15:0]
adc_ext_trg0	tim1_oc1
adc_ext_trg1	tim1_oc2
adc_ext_trg2	tim1_oc3
adc_ext_trg3	tim2_oc2
adc_ext_trg4	tim3_trgo
adc_ext_trg5	tim4_oc4
adc_ext_trg6	exti11
adc_ext_trg7	Reserved
adc_ext_trg8	Reserved
adc_ext_trg9	tim1_trgo
adc_ext_trg10	tim1_trgo2
adc_ext_trg11	tim2_trgo
adc_ext_trg12	tim4_trgo
adc_ext_trg13	tim6_trgo
adc_ext_trg14	tim15_trgo
adc_ext_trg15	tim3_oc4
adc_ext_trg16	exti15
adc_ext_trg17	Reserved
adc_ext_trg18	lptim1_ch1
adc_ext_trg19	lptim2_ch1
adc_ext_trg20	lptim3_ch1
adc_ext_trg21	lptim4_out
adc_ext_trg22	Reserved
adc_ext_trg23	Reserved
adc_ext_trg24	Reserved
adc_ext_trg25	Reserved
adc_ext_trg26	Reserved
adc_ext_trg27	Reserved
adc_ext_trg28	Reserved
adc_ext_trg29	Reserved
adc_ext_trg30	Reserved
adc_ext_trg31	Reserved
adc_jext_trg0	tim1_trgo

Table 178. ADC1/2 interconnection (continued)

Signal name	Source/destination
adc_jext_trg1	tim1_oc4
adc_jext_trg2	tim2_trgo
adc_jext_trg3	tim2_oc1
adc_jext_trg4	tim3_oc4
adc_jext_trg5	tim4_trgo
adc_jext_trg6	exti15
adc_jext_trg7	Reserved
adc_jext_trg8	tim1_trgo2
adc_jext_trg9	Reserved
adc_jext_trg10	Reserved
adc_jext_trg11	tim3_oc3
adc_jext_trg12	tim3_trgo
adc_jext_trg13	tim3_oc1
adc_jext_trg14	tim6_trgo
adc_jext_trg15	tim15_trgo
adc_jext_trg16	Reserved
adc_jext_trg17	Reserved
adc_jext_trg18	lptim1_ch2
adc_jext_trg19	lptim2_ch2
adc_jext_trg20	lptim3_ch1
adc_jext_trg21	lptim4_out
adc_jext_trg22	Reserved
adc_jext_trg23	Reserved
adc_jext_trg24	Reserved
adc_jext_trg25	Reserved
adc_jext_trg26	Reserved
adc_jext_trg27	Reserved
adc_jext_trg28	Reserved
adc_jext_trg29	Reserved
adc_jext_trg30	Reserved
adc_jext_trg31	Reserved

22.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 120: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

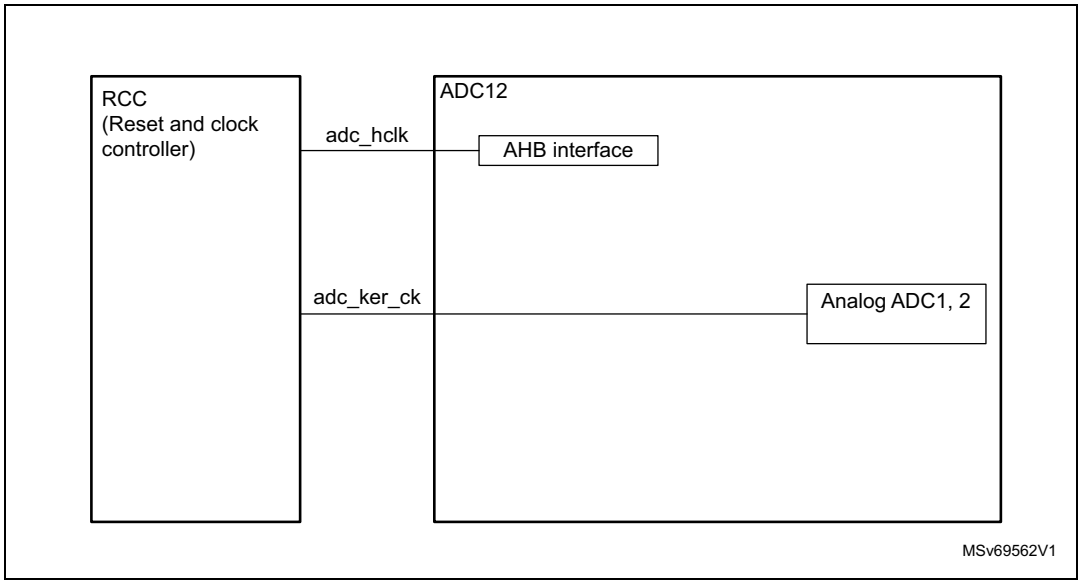
1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 120. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

22.4.4 ADC connectivity

ADC inputs are connected to external channels as well as internal sources as described below.

Figure 121. ADC1 connectivity

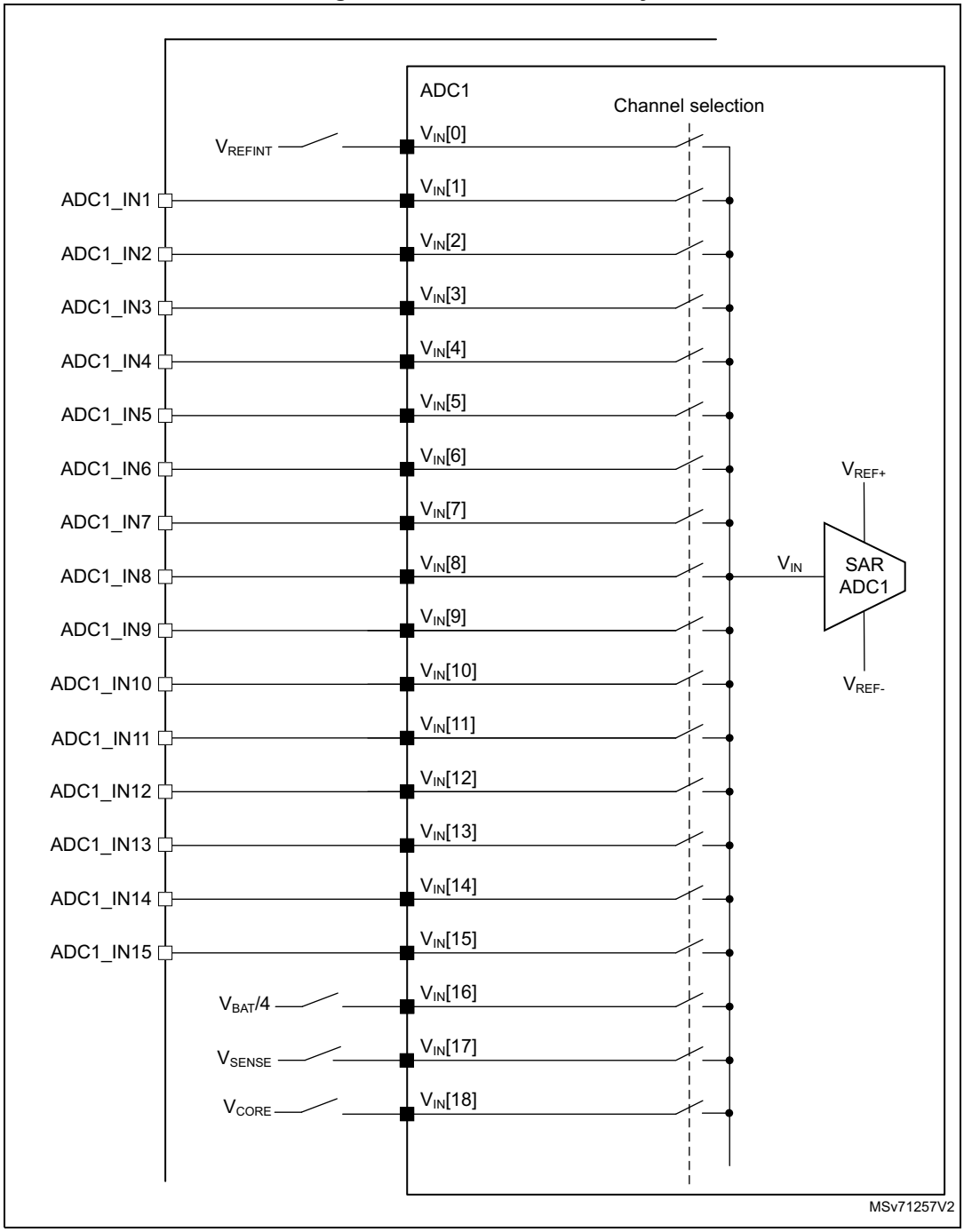
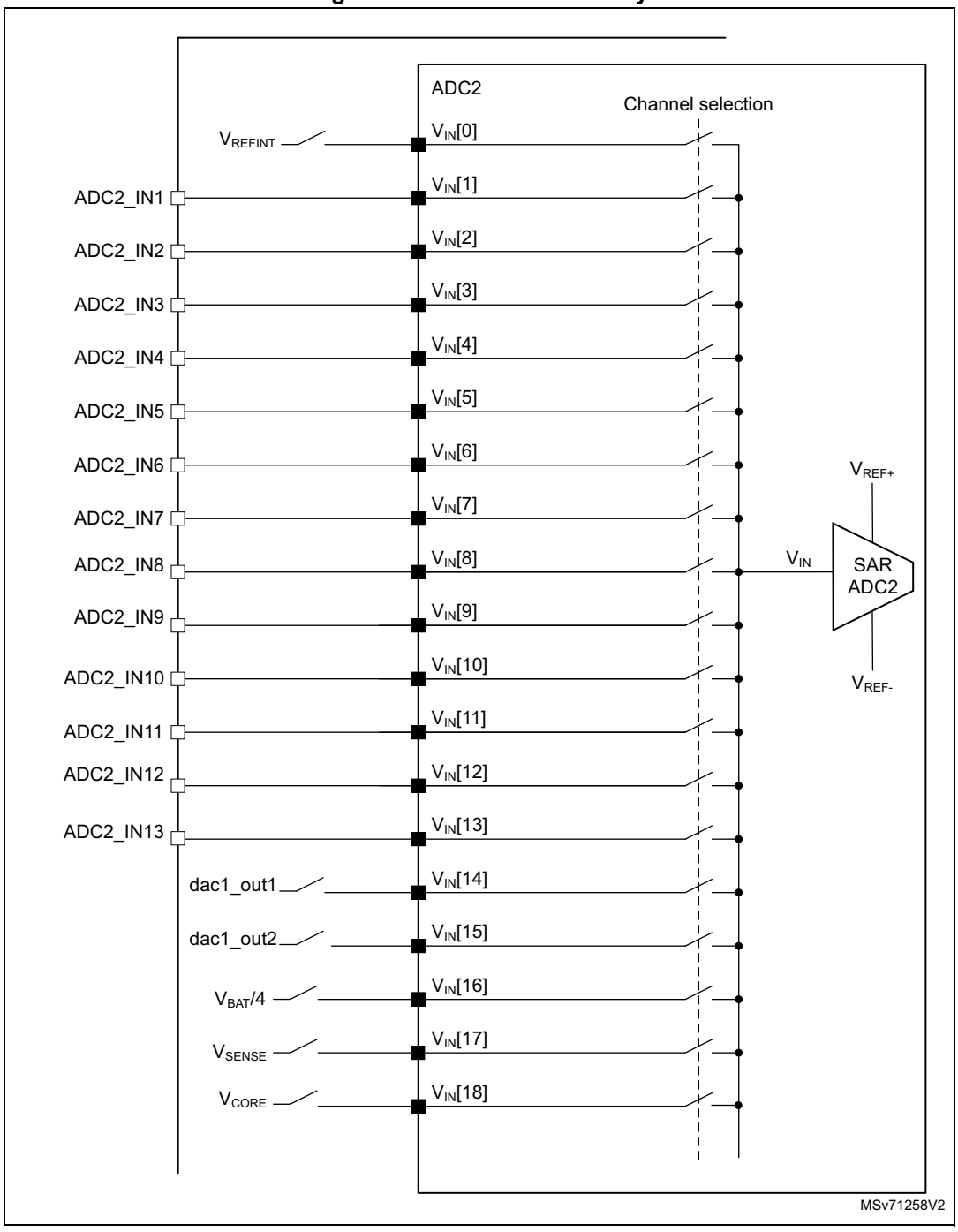


Figure 122. ADC2 connectivity



22.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

22.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of the DEEPPWD bit is 1 in the ADC_CR register).

To start ADC operations, it is first required to exit Deep-power-down mode by clearing the DEEPPWD bit.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the ADVREGEN bit of the ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software. For the startup time of the ADC voltage regulator, refer to the device datasheet for $T_{ADCVREG_STUP}$ parameter.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of ADC_CR. Power can then be saved by disabling the ADC voltage regulator (by clearing ADVREGEN).

Additional power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

Note: Setting DEEPPWD automatically disables the ADC voltage regulator and the ADVREGEN bit is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN cleared), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or reapply the calibration factor, which was previously saved (refer to *Section Calibration*).

22.4.7 Calibration (ADCAL)

A software calibration is required before launching the ADC. During the calibration procedure, the ADC calculates a calibration factor, which is applied internally to the ADC until the next ADC power-off. During the calibration, the application must not use the ADC and must wait until the calibration is complete.

The calibration is preliminary to any ADC operation. It removes the offset error, which may vary from chip to chip due to process, voltage and temperature variation.

The calibration is initiated by software by setting the ADCAL bit. It can be initiated only when all the following conditions are met:

- The ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1).
- The ADC is disabled (ADEN = 0).

The ADCAL bit stays at 1 during all the calibration sequence. It is then deasserted by hardware as soon the calibration completes. At this time, the calibration factor can be read from the bits 6 to 0 of the ADC_DR register. The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, when the ADC operating conditions change (the main contributor to ADC offset variations is V_{DDA}, and the temperature to a lesser extent), rerun a calibration cycle. It is recommended to recalibrate the ADC when the V_{DDA} voltage changed more than 10%.

The internal analog calibration is lost in the following cases:

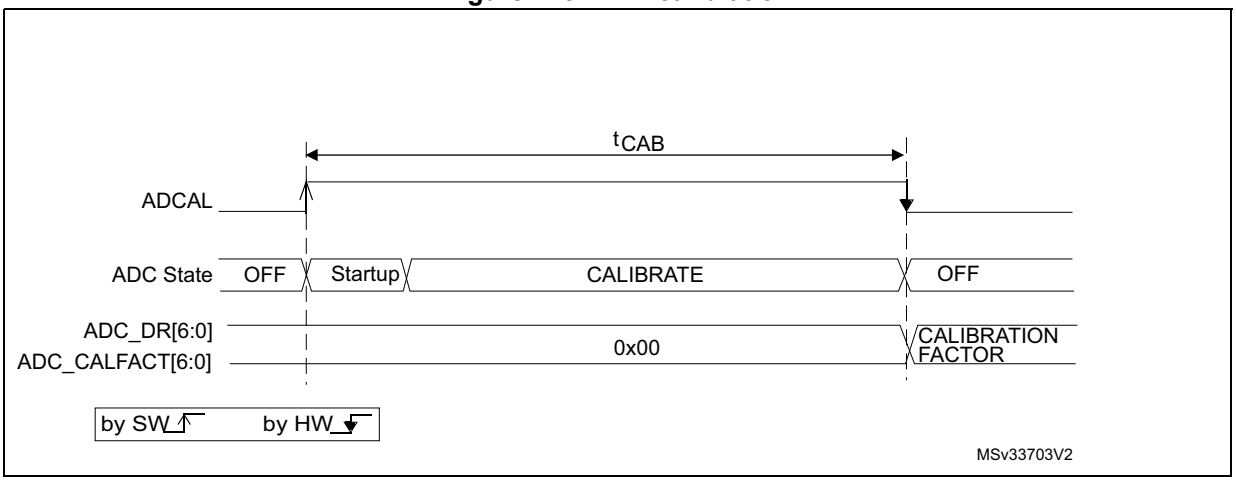
- The ADC power is removed (for example, when the product enters standby mode). Still, to avoid spending time recalibrating the ADC, it is possible to save and restore the calibration factor by software (provided the temperature and voltage are stable during the ADC power-down).
- The ADC is reset.

The calibration factor can be written if the ADC is enabled but no conversions are ongoing (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software procedure to calibrate the ADC

1. Ensure ADEN = 0, ADVREGEN = 1, AUTOFF = 0, DEEPPWD = 0, and DMAEN = 0.
2. Set ADCAL.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit of the ADC_IER register.
4. The calibration factor can be read from bits 6 to 0 of the ADC_DR or ADC_CALFACT register.

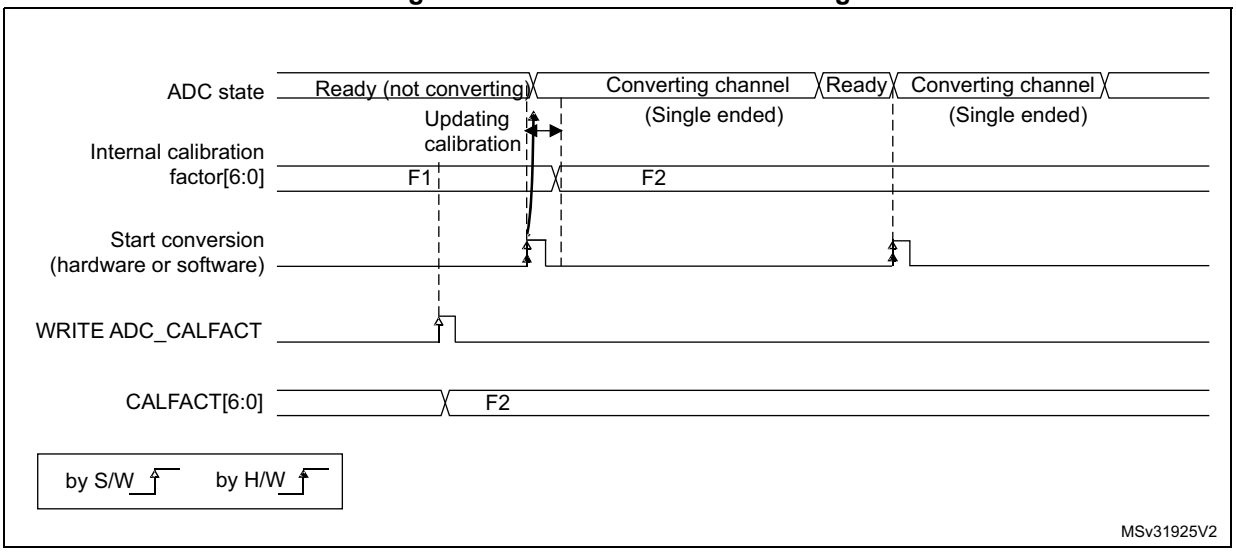
Figure 123. ADC calibration



Software procedure to force the calibration factor

1. Ensure ADEN and ADSTART = 0 (ADC started with no conversion ongoing).
2. Program the saved calibration factor to the ADC_CALFACT register.
3. The calibration factor is used as soon as a new conversion is launched.

Figure 124. Calibration factor forcing



22.4.8 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure described in *Section 22.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD is cleared and ADVREGEN is set, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 125*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 22.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

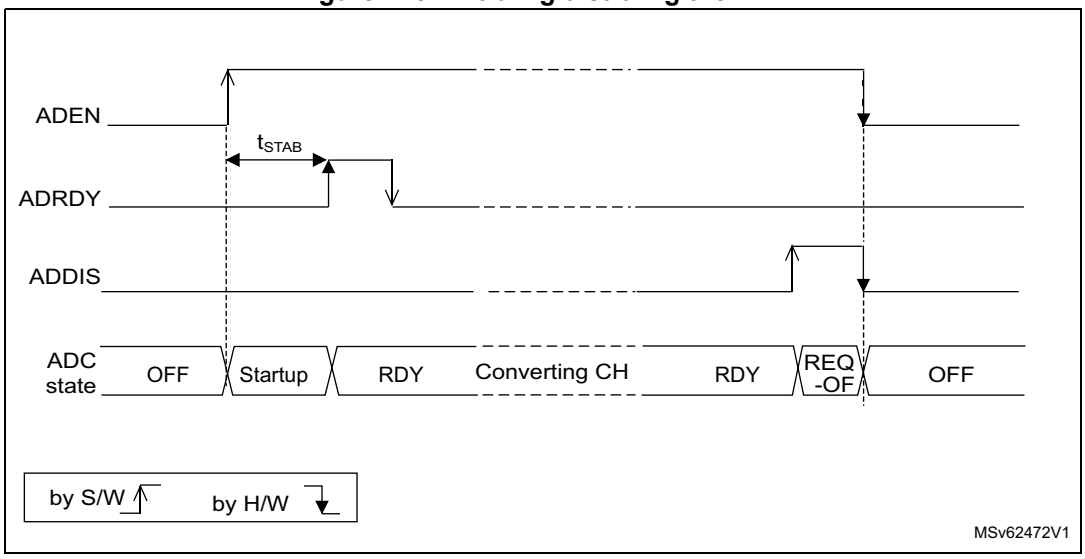
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 125. Enabling/disabling the ADC



22.4.9 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can then write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 22.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for details).
- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC. In addition, the software can write the ADC_JSQR register only if JADSTART is cleared and no injected conversion is ongoing. Refer to *Section 22.7.14: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: There is no hardware protection to prevent forbidden write accesses to ADC control bits. In such cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

22.4.10 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 19 multiplexed channels per ADC, out of which:

- Up to 15 analog inputs coming from GPIO pads (ADCx_IN[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 6 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - Internal temperature sensor (V_{SENSE})
  - V_{BAT} monitoring channel ($V_{BAT}/4$)
  - V_{CORE} voltage
  - DAC internal channel

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, TSEN, VBATEN in the ADCC_CCR registers, or by enabling the corresponding bit in the *ADC option register (ADC_OR)*.

Refer to *ADC interconnection* tables in *Section 22.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_IN3, ADCx_IN8, ADCx_IN2, ADCx_IN2, ADCx_IN0, ADCx_IN2, and ADCx_IN2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The

total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 22.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

In addition, the ADC_JSQR register must not be modified while an injected conversion is ongoing. ADC injected conversions must consequently be stopped by setting JADSTP, except when JQDIS is cleared.

22.4.11 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{IN} of each ADC input. For example, if the ADC converts ADCx_IN1, PCSEL1 bit must also be set in ADC_PCSEL register.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

22.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 6.5 ADC clock cycles
- SMP = 011: 11.5 ADC clock cycles
- SMP = 100: 23.5 ADC clock cycles
- SMP = 101: 46.5 ADC clock cycles
- SMP = 110: 246.5 ADC clock cycles
- SMP = 111: 1499.5 ADC clock cycles

The total conversion time is calculated as follows:

$$T_{CONV} = \text{Sampling time} + 12.5 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 30 MHz and a sampling time of 2.5 ADC clock cycles:

$$T_{CONV} = (2.5 + 12.5) \times \text{ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

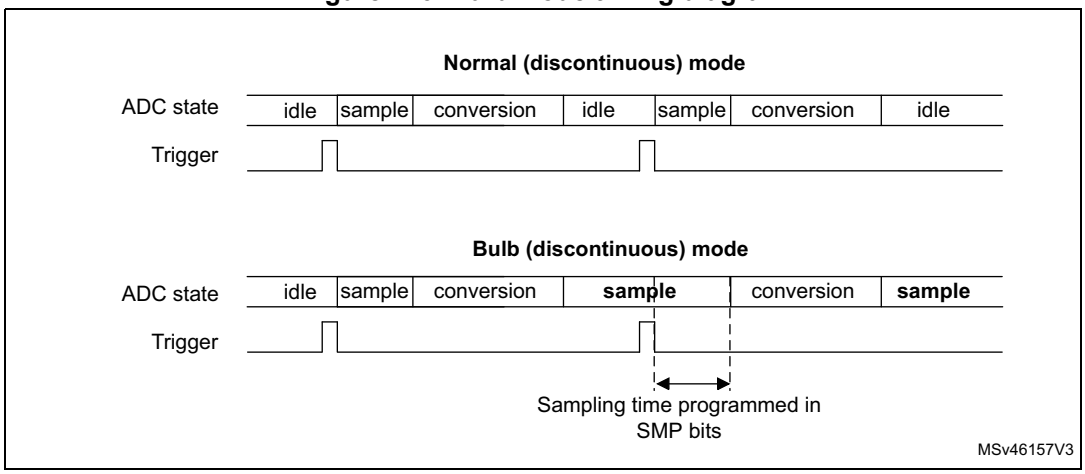
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 126. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and injected conversions are not supported.

The sampling time control trigger mode is not compatible with the following dual modes: DUAL[4:0] = 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, and 0b01001.

22.4.13 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

The single conversion mode is compatible with bulb and sampling time control trigger modes.

Note: To convert a single channel, program a sequence with a length of 1.

22.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

22.4.15 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT and EXTEN don't care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion) or at the end of each subgroup processing if JDISCEN = 1
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

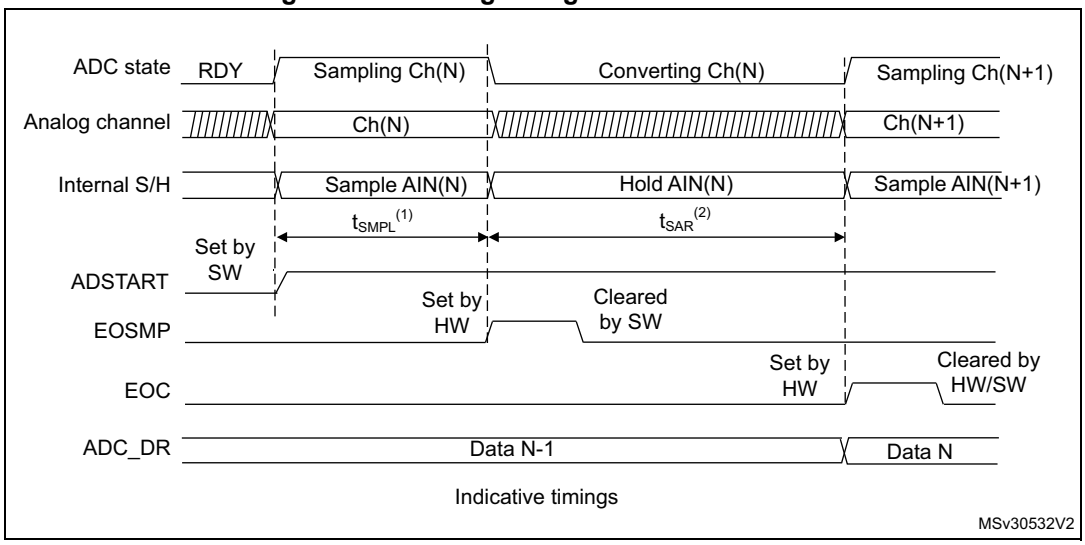
22.4.16 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ _{|min} + 12.5\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 83.333\ ns\ _{|min} + 416.667\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 30$ MHz)

Figure 127. Analog-to-digital conversion time



1. t_{SMPL} depends on SMP[2:0].
2. t_{SAR} depends on RES[2:0].
3. The bulb and the sampling time control trigger modes are not described in the above figure.

22.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 128. Stopping ongoing regular conversions

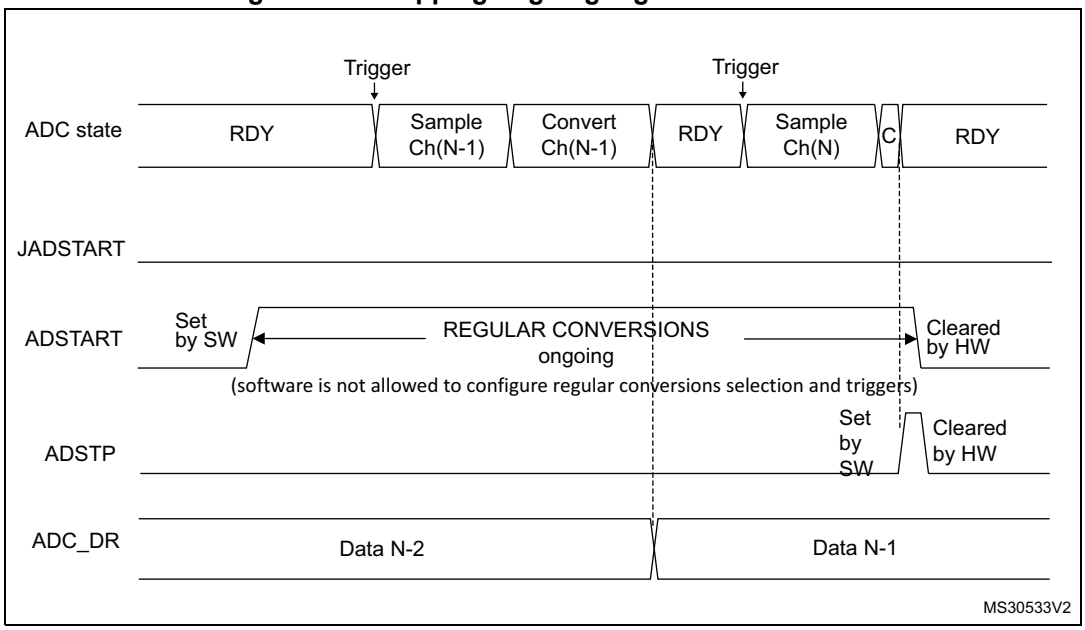
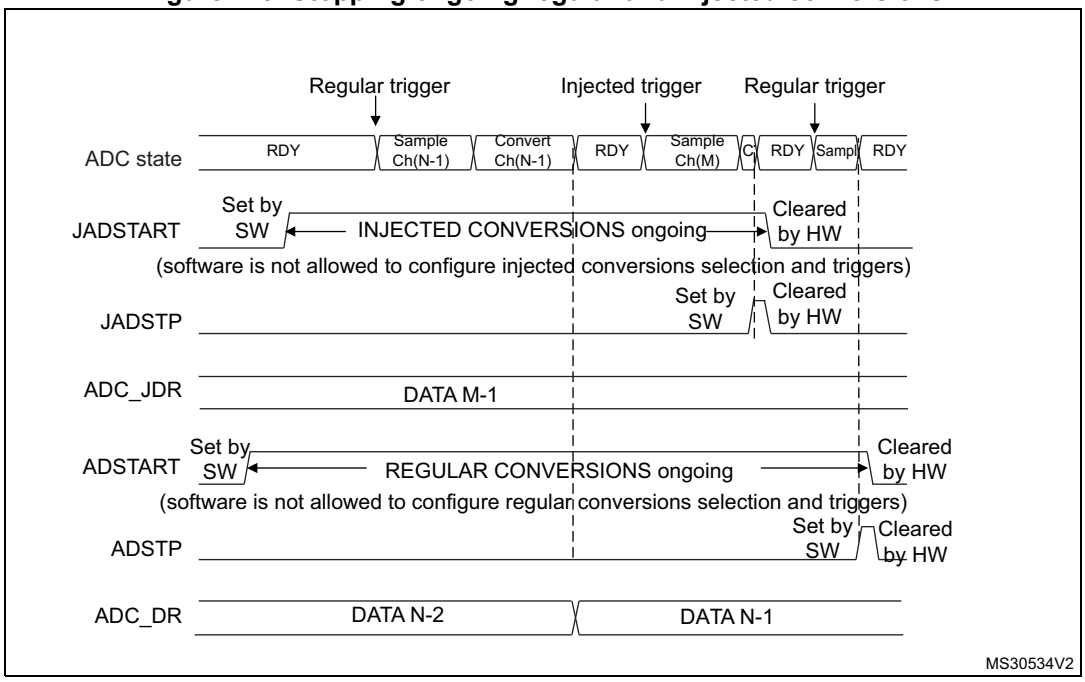


Figure 129. Stopping ongoing regular and injected conversions



22.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 179 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 179. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in hardware trigger mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

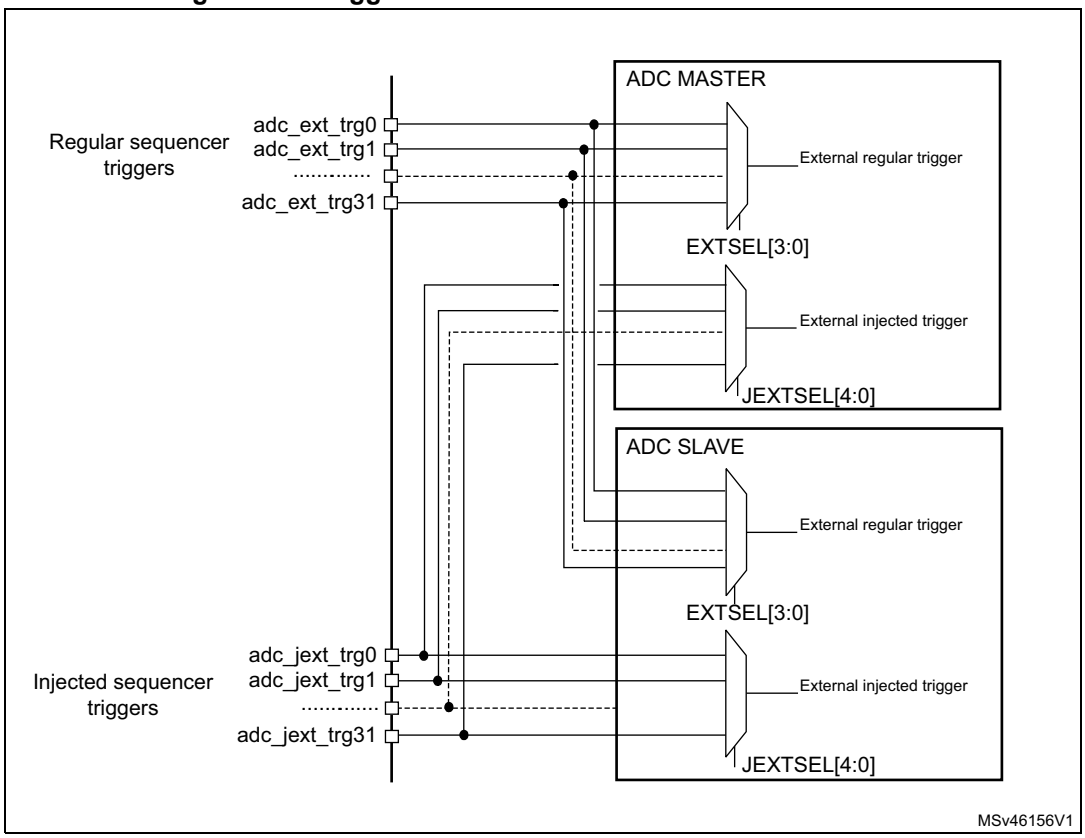
Table 180. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Figure 130. Triggers shared between ADC master and slave



Refer to *ADC interconnection* tables in *Section 22.4.2: ADC pins and internal signals* for the list of the external triggers that can be used for regular conversion.

22.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 131* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 28 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 29 ADC clock cycles.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

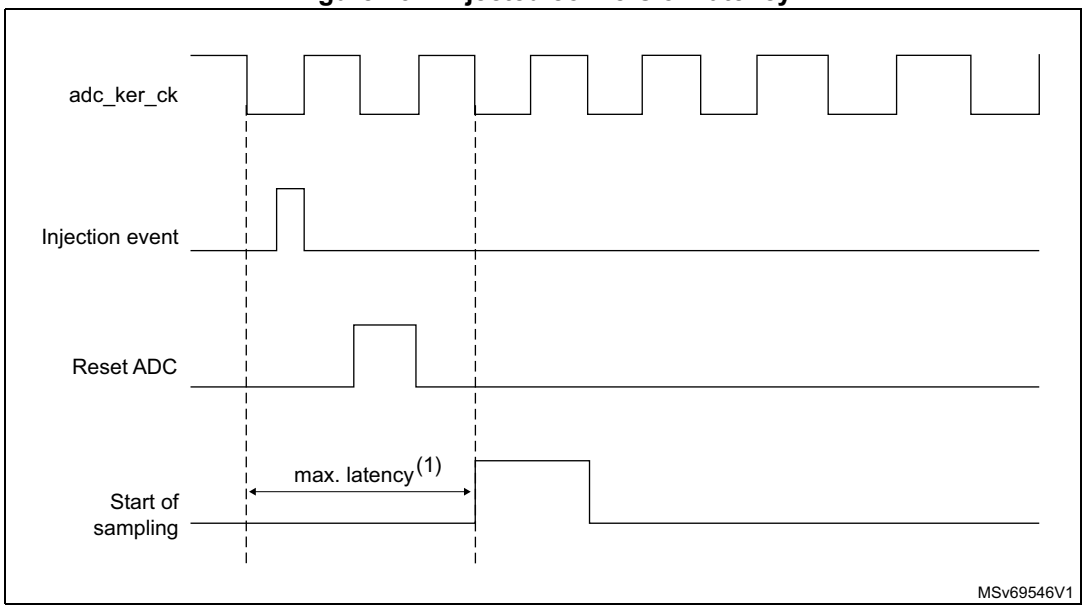
In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the autoinjected and discontinuous modes simultaneously.

Figure 131. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

22.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11

  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.
  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

22.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to two contexts for the next injected sequence of conversions. JQDIS bit of the ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in the ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in the ADC_JSQR register)

All the parameters of the context are defined in a single register ADC_JSQR and this register implements a queue of two buffers, allowing the bufferization of up to two sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written to the ADC_JSQR register is stored into the queue of context.
- At the beginning, the queue is empty and the first write access to the ADC_JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the queue is consumed and the context changes according to the next ADC_JSQR parameters stored in the queue. This new context is applied for the next injected sequence of conversions.
- A queue overflow occurs when writing to the ADC_JSQR register while the queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of ADC_JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the queue becomes empty, depending on the value of the JQM control bit of the ADC_CFGR register:
  - If JQM = 0, the queue is empty just after enabling the ADC, but then it can never be empty during run operations: the queue always maintains the last active

context and any further valid start of the injected sequence is served according to the last active context.

  - If JQM = 1, the queue can be empty after the end of an injected sequence or if the queue is flushed. When this occurs, there is no more context in the queue and hardware writes a new injected context into the ADC_JSQR register.
- Reading the ADC_JSQR register returns the current the ADC_JSQR context which is active at that moment. When the ADC_JSQR context is empty, the JSQi is read as 0x00.
- The queue is flushed when stopping injected conversions by setting JADSTP or when disabling the ADC by setting ADDIS:
  - If JQM = 0, the queue is maintained with the last active context.
  - If JQM = 1, the queue becomes empty and triggers are ignored.

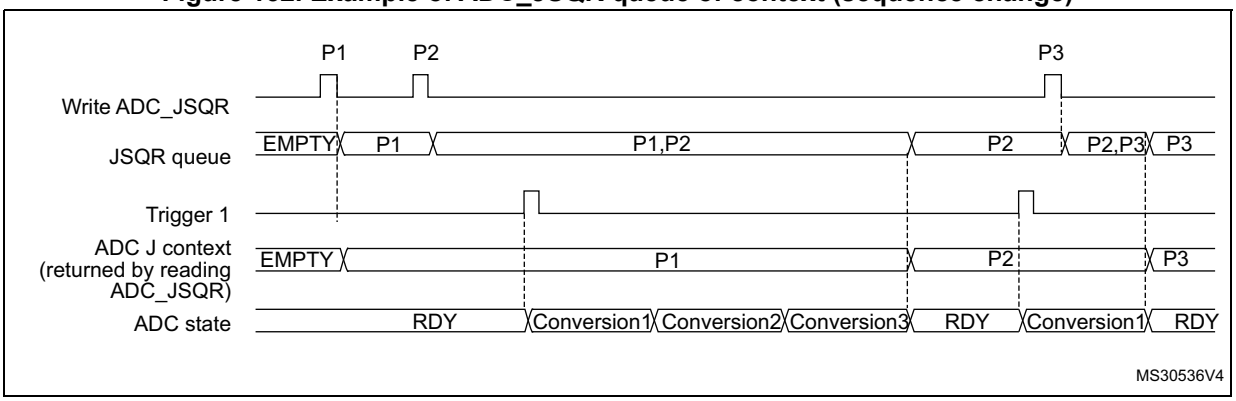
Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the queue. The first trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *First trigger, discontinuous. Sequence 1: context 1 consumed, first conversion carried out*
- *Second trigger, disc. Sequence 1: Second conversion.*
- *Third trigger, discontinuous. Sequence 1: Third conversion.*
- *Fourth trigger, discontinuous. Sequence 2: context 2 consumed, first conversion carried out.*
- *Fifth trigger, discontinuous. Sequence 2: Second conversion.*
- *Sixth trigger, discontinuous. Sequence 2: Third conversion.*

Behavior when changing the trigger or sequence context

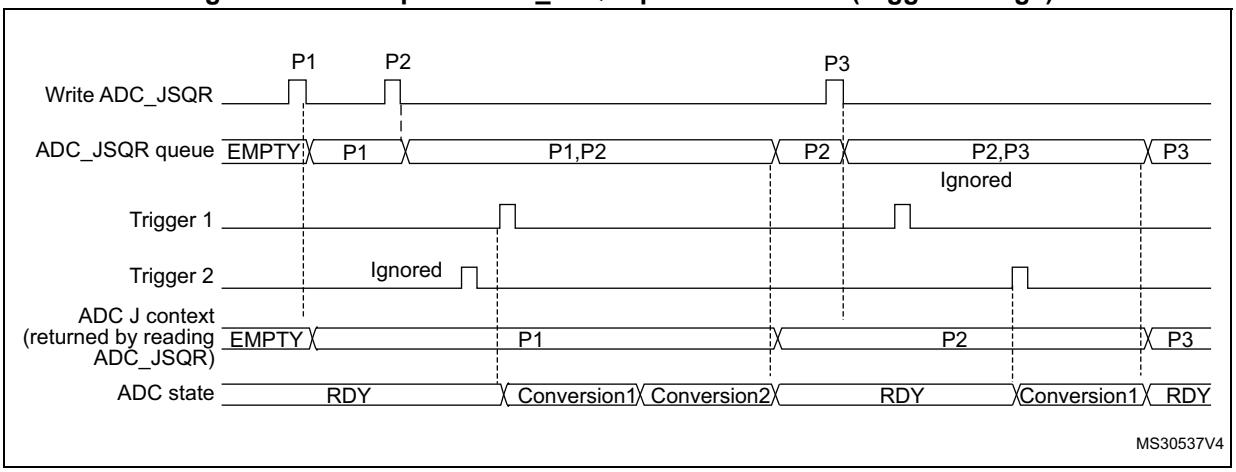
Figure 132 and *Figure 133* show the behavior of the context queue when changing the sequence or the triggers.

Figure 132. Example of ADC_JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 133. Example of ADC_JSQR queue of context (trigger change)

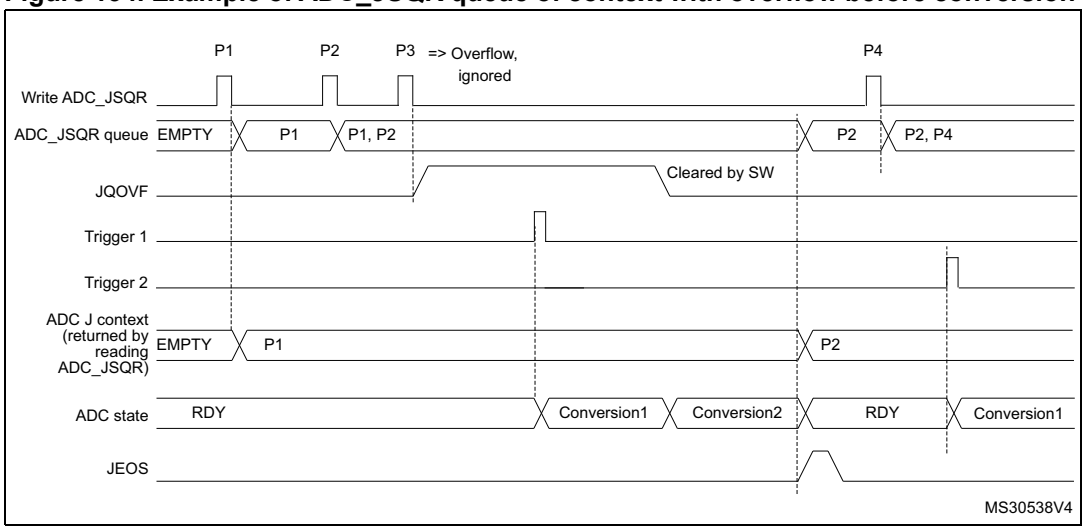


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

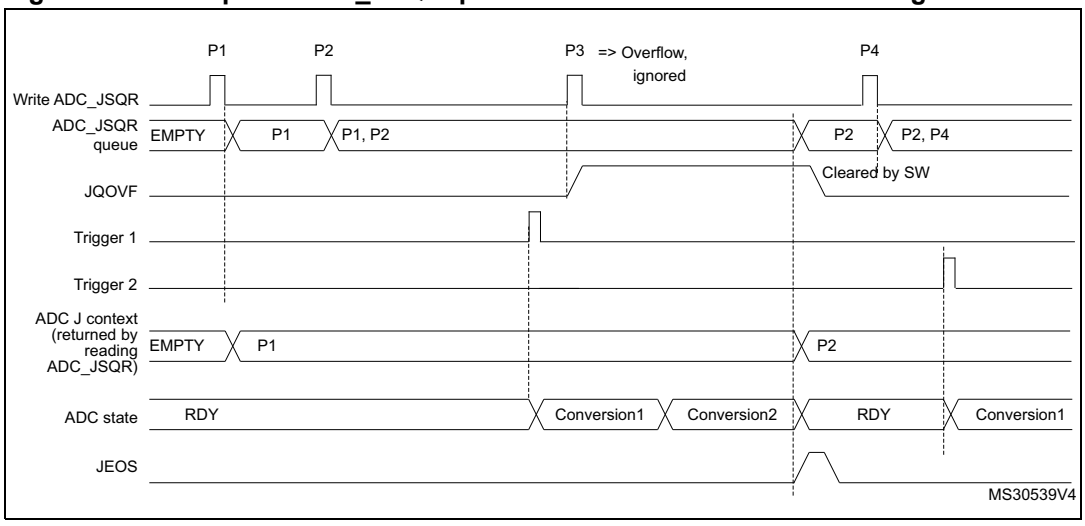
The *Figure 134* and *Figure 135* show the behavior of the context queue if an overflow occurs before or during a conversion.

Figure 134. Example of ADC_JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 135. Example of ADC_JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

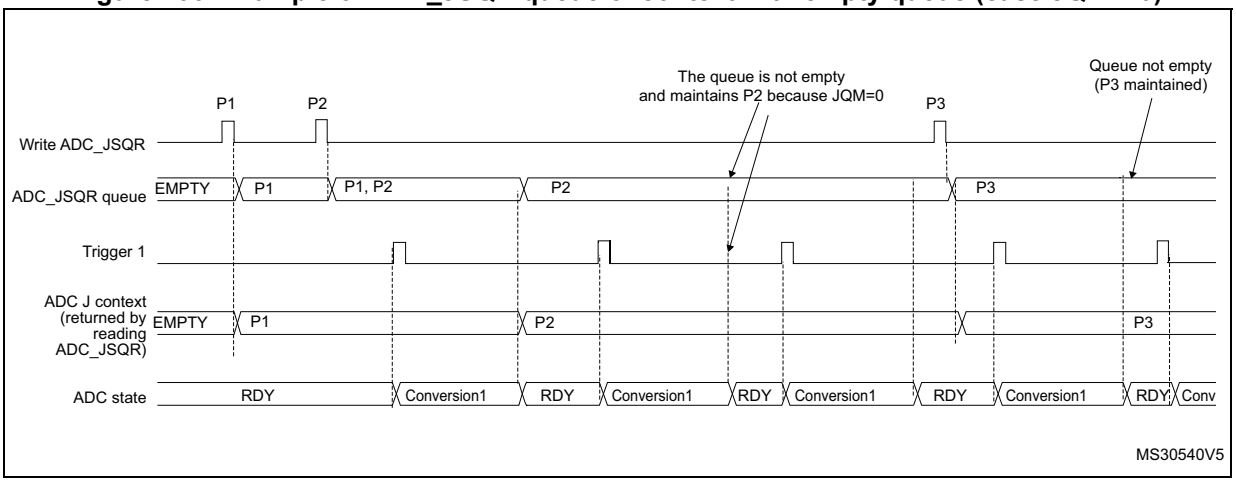
It is recommended to manage the queue overflows as described below:

- After each P context write to the ADC_JSQR register, the JQOVF flag shows if the write has been ignored or not (an interrupt can be generated).
- Avoid queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 136 and *Figure 137* show the behavior of the context queue when the queue becomes empty in both cases JQM = 0 or 1.

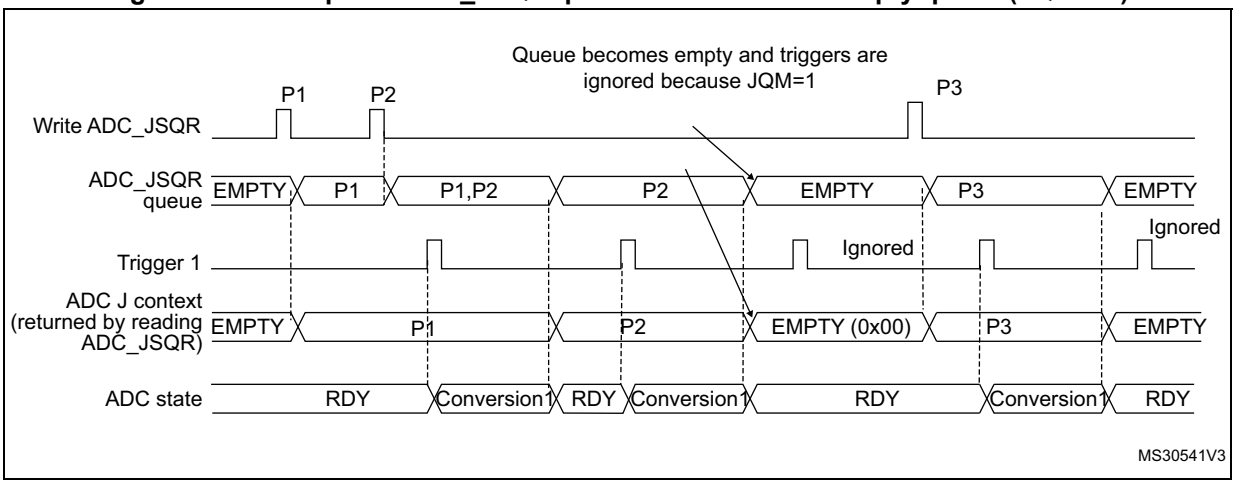
Figure 136. Example of ADC_JSQR queue of context with empty queue (case JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 137. Example of ADC_JSQR queue of context with empty queue (JQM = 1)

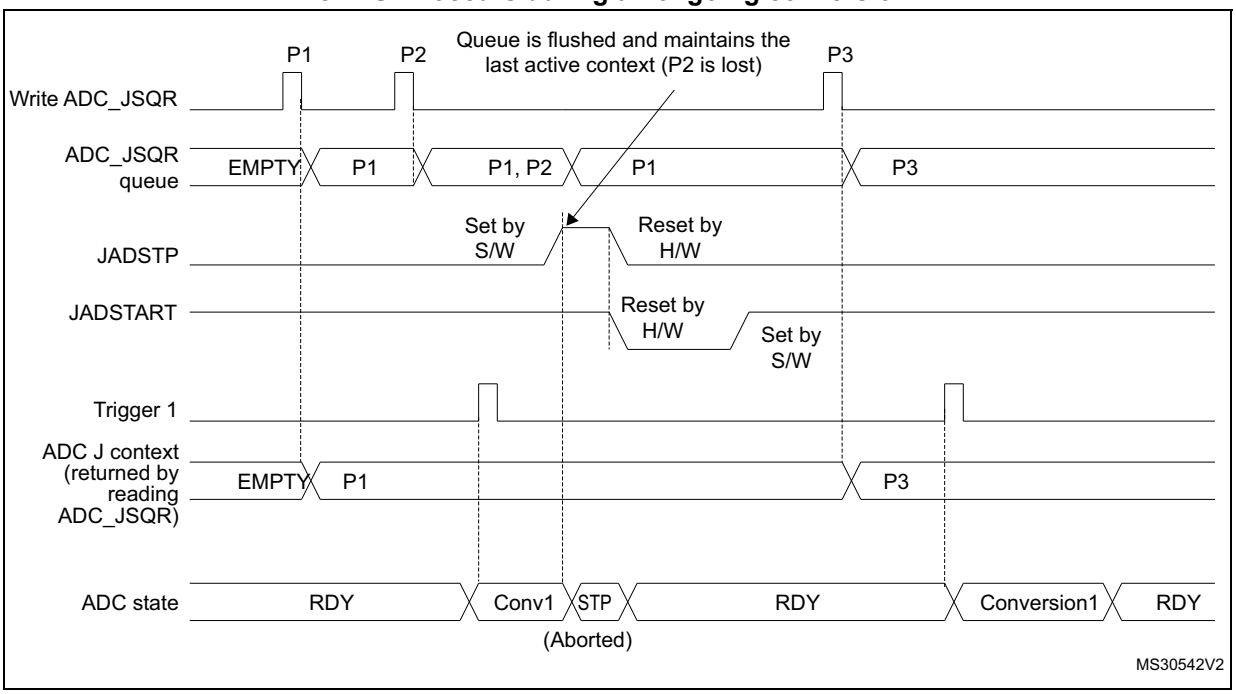


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

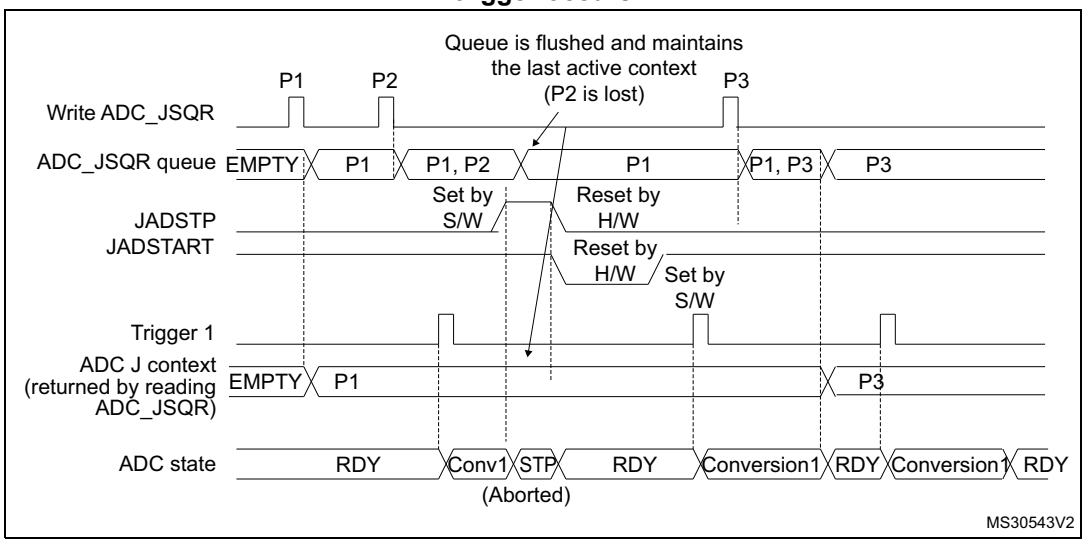
The figures below show the behavior of the context queue in various situations when the queue is flushed.

Figure 138. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



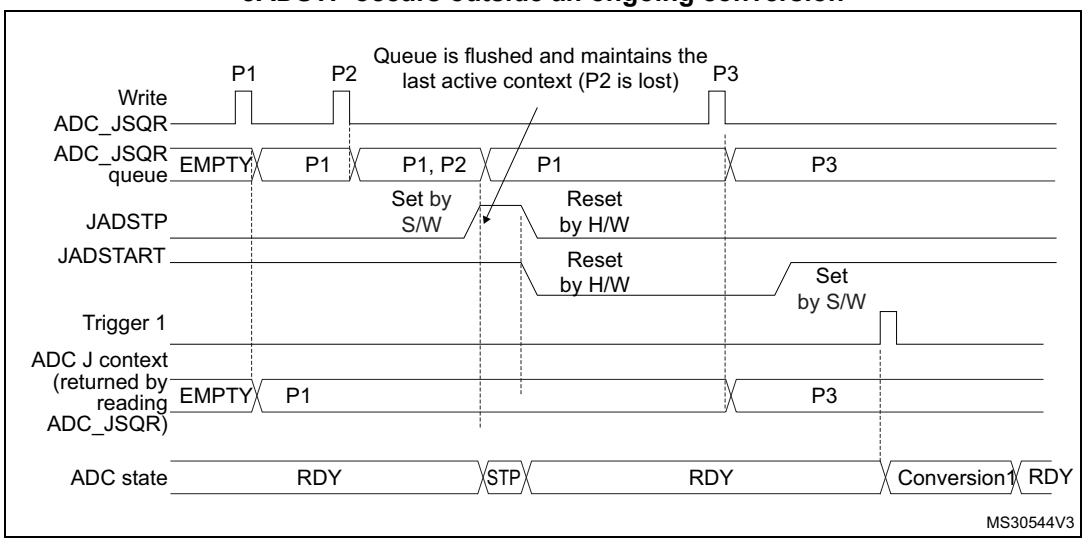
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 139. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



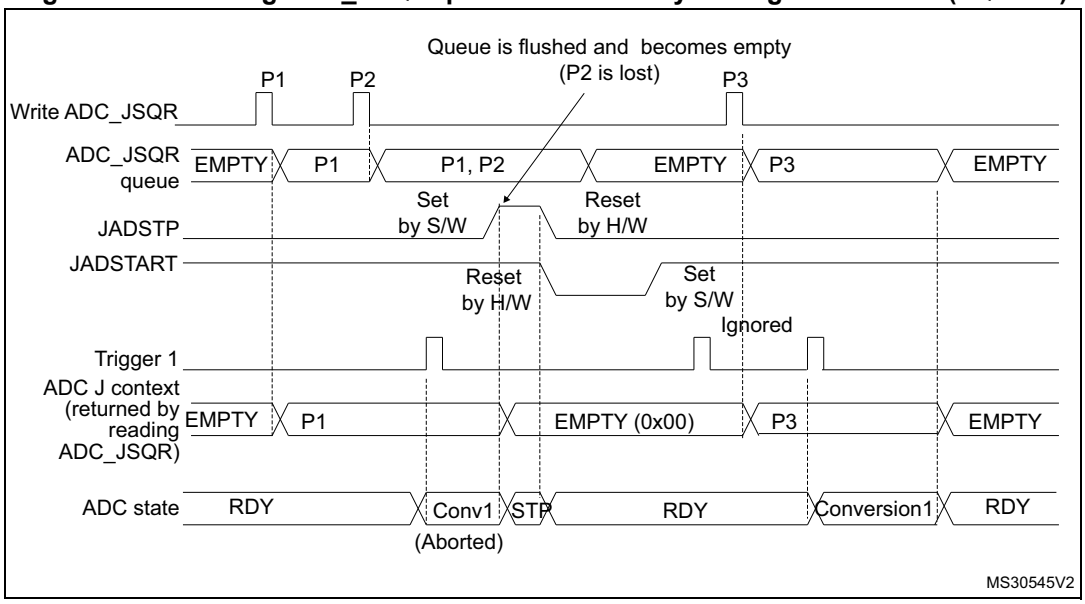
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 140. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



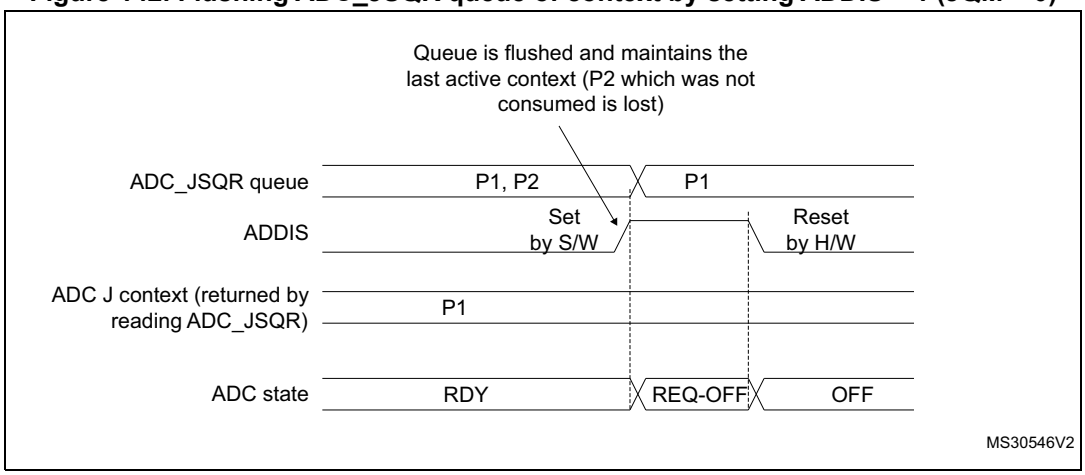
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 141. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1)



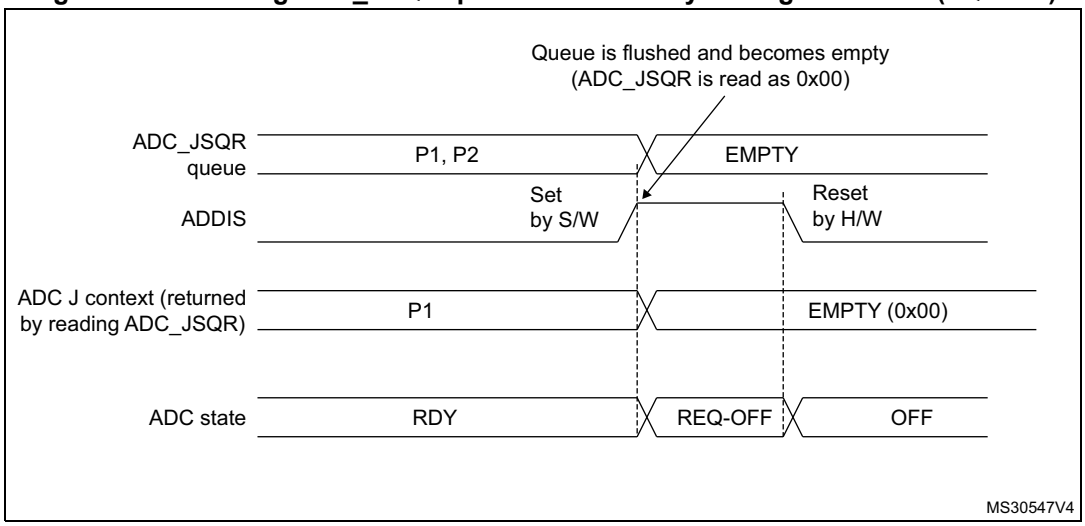
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 142. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 143. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start an ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when the JQM bit is reset:

1. Write a dummy ADC_JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
2. Set JADSTART
3. Set JADSTP
4. Wait until JADSTART is reset
5. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS of the ADC_CFGR register.

22.4.22 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 148*, *Figure 149*, *Figure 150* and *Figure 151* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to .

Table 181. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 2.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 30 MHz
12	12.5 ADC clock cycles	416.667 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	283.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300 ns

22.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

22.4.24 End of conversion sequence (EOS, JEOS)

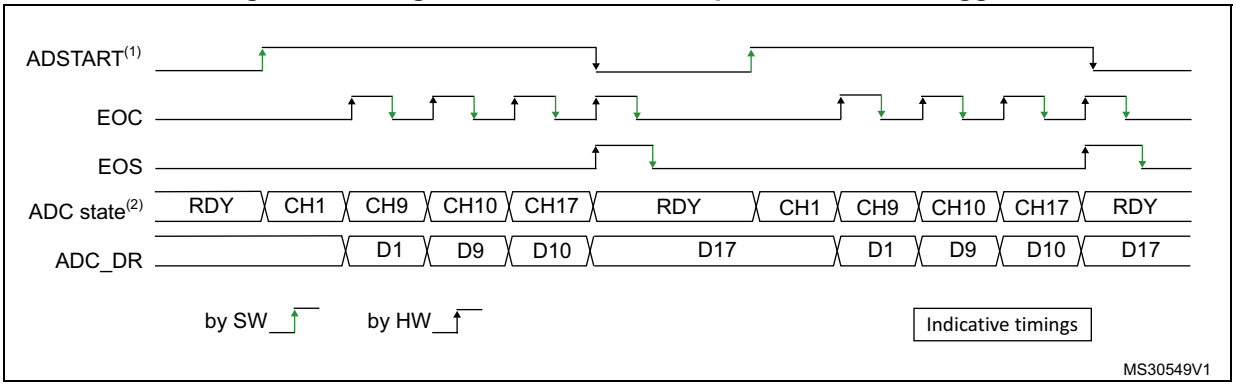
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

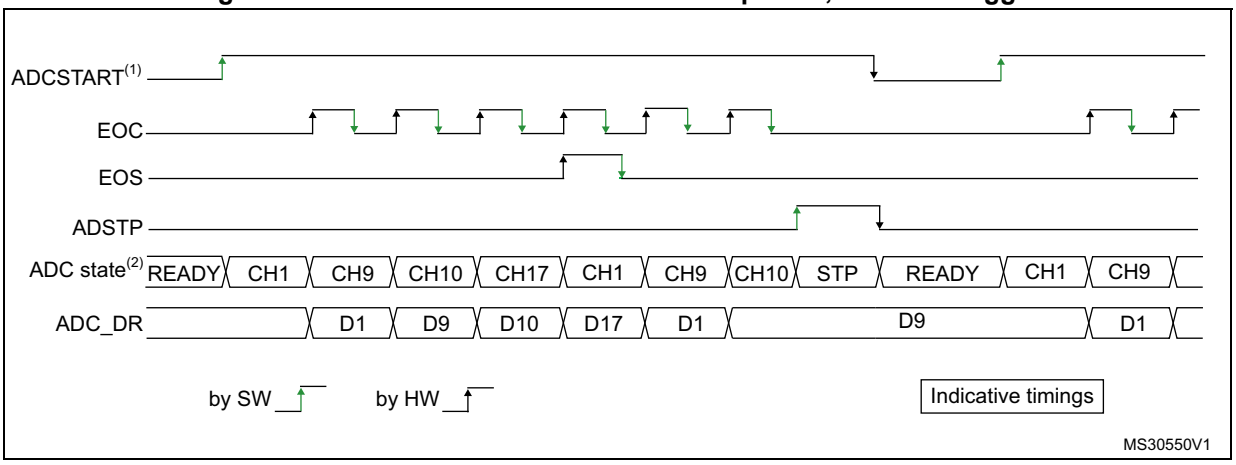
22.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 144. Single conversions of a sequence, software trigger



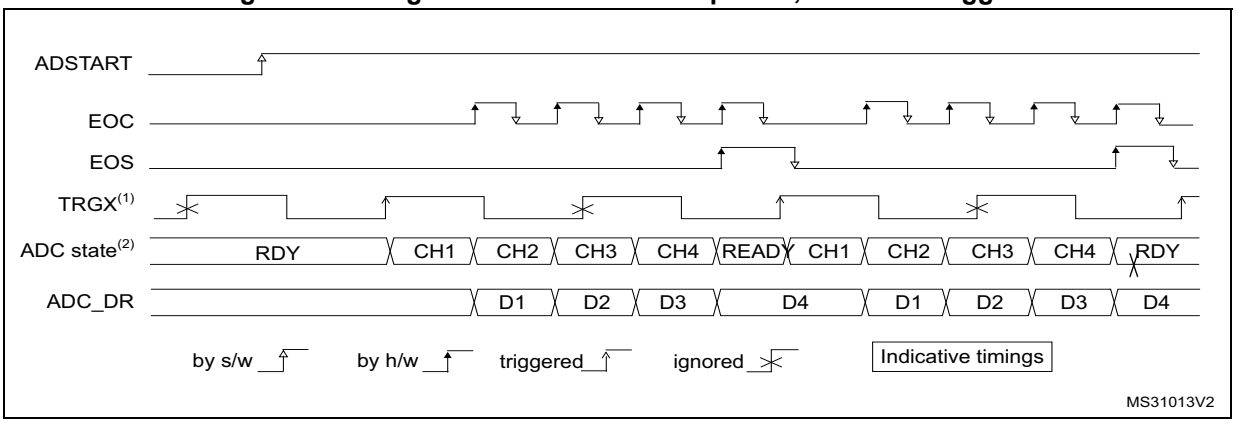
1. EXTEN[1:0] = 00, CONT=0
2. Channels selected = 1, 9, 10, 17; AUTDLY=0.

Figure 145. Continuous conversion of a sequence, software trigger



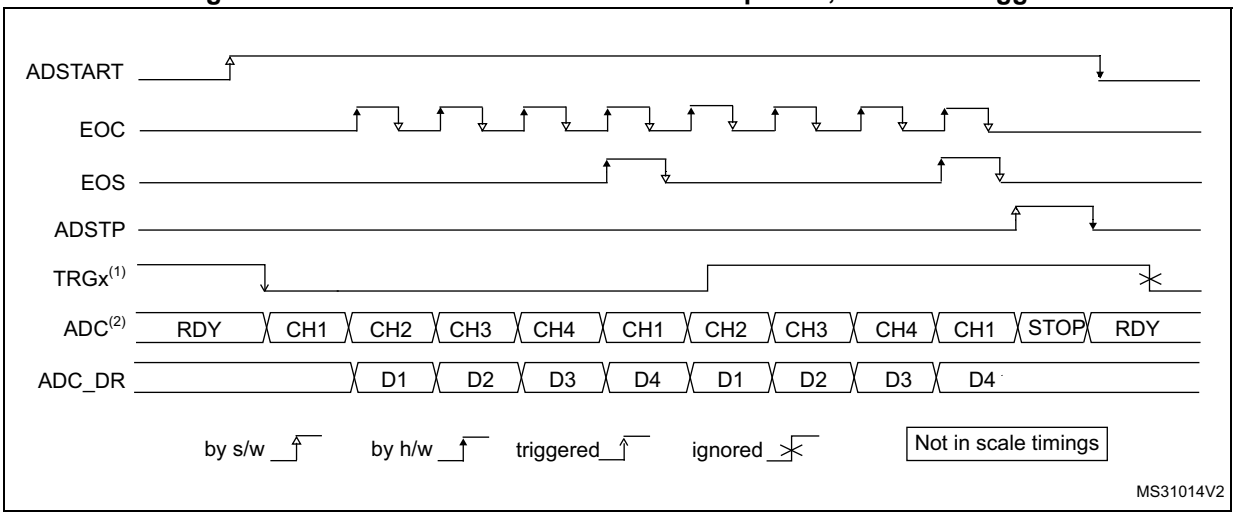
1. EXTEN[1:0] = 00, CONT=1
2. Channels selected = 1, 9, 10, 17; AUTDLY=0.

Figure 146. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 147. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

22.4.26 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO. Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0] and LSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 148*, *Figure 149*, *Figure 150* and *Figure 151* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: It is not allowed to select the same channel for different OFFSETy_CH[4:0] values except when OFFSET[21:0] is 0.

Table 186 describes how the computation is performed for all the possible ADC resolutions and offset settings.

Table 182. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 148, *Figure 149*, *Figure 150* and *Figure 151* show alignments for signed and unsigned data.

Figure 148. Right alignment (offset disabled, unsigned value)

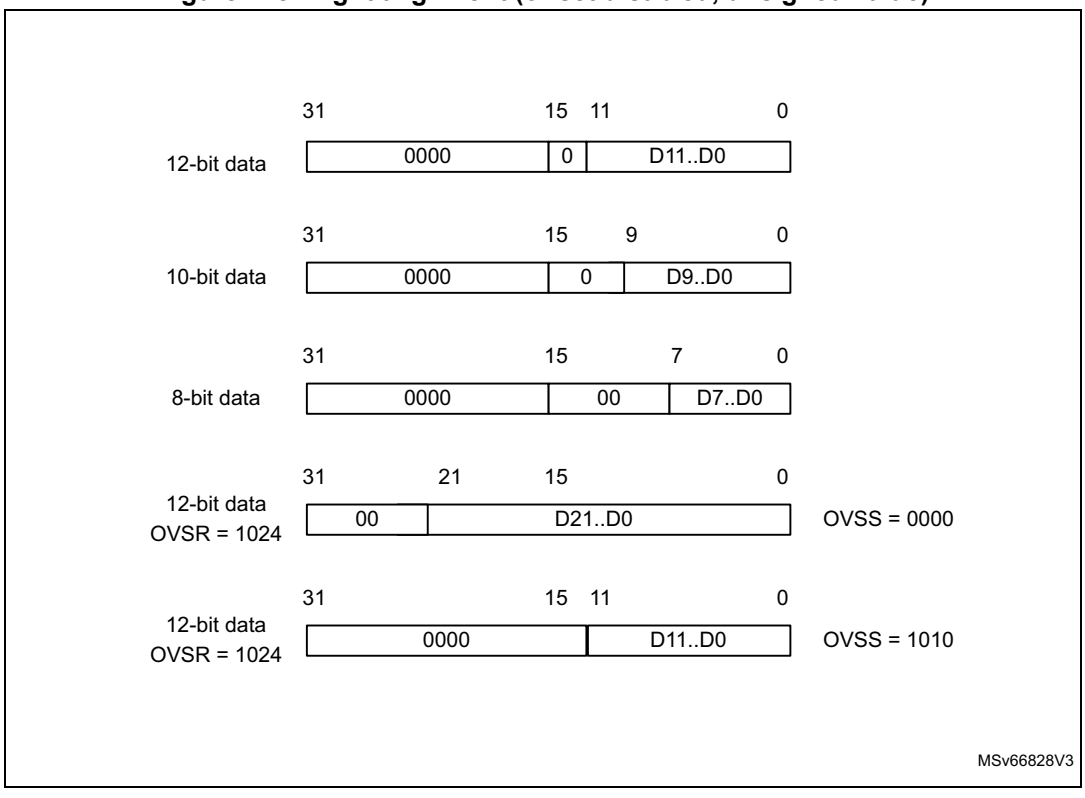


Figure 149. Right alignment (offset enabled, signed value)

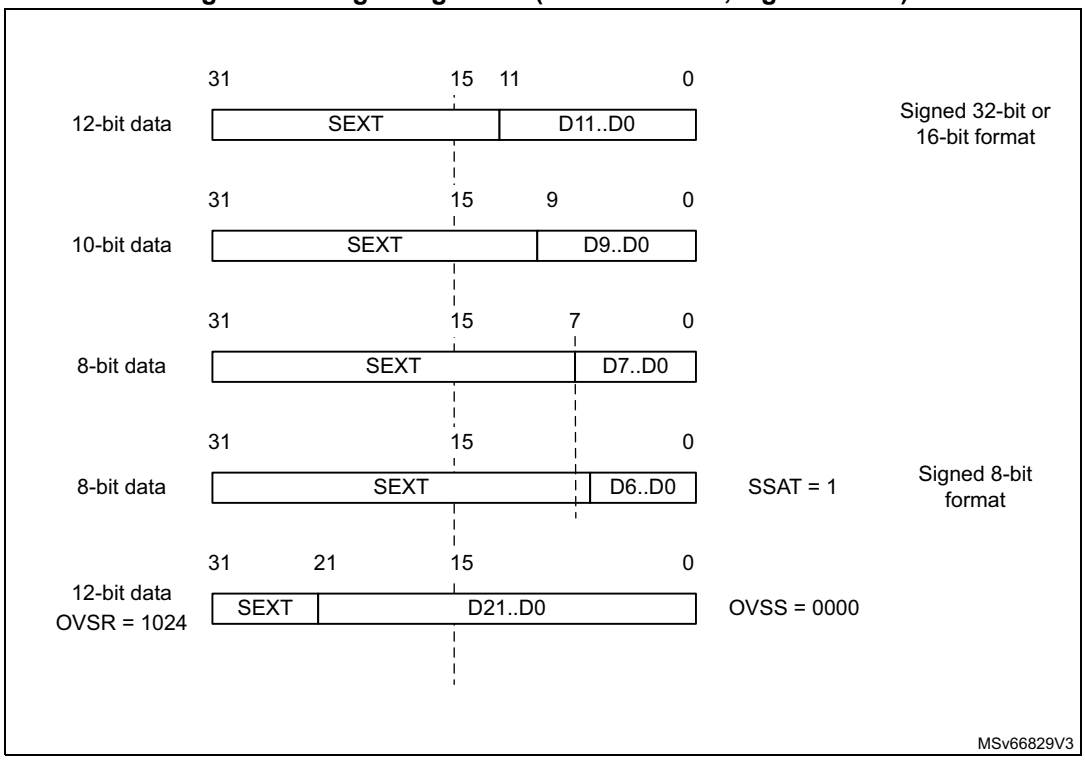


Figure 150. Left alignment (offset disabled, unsigned value)

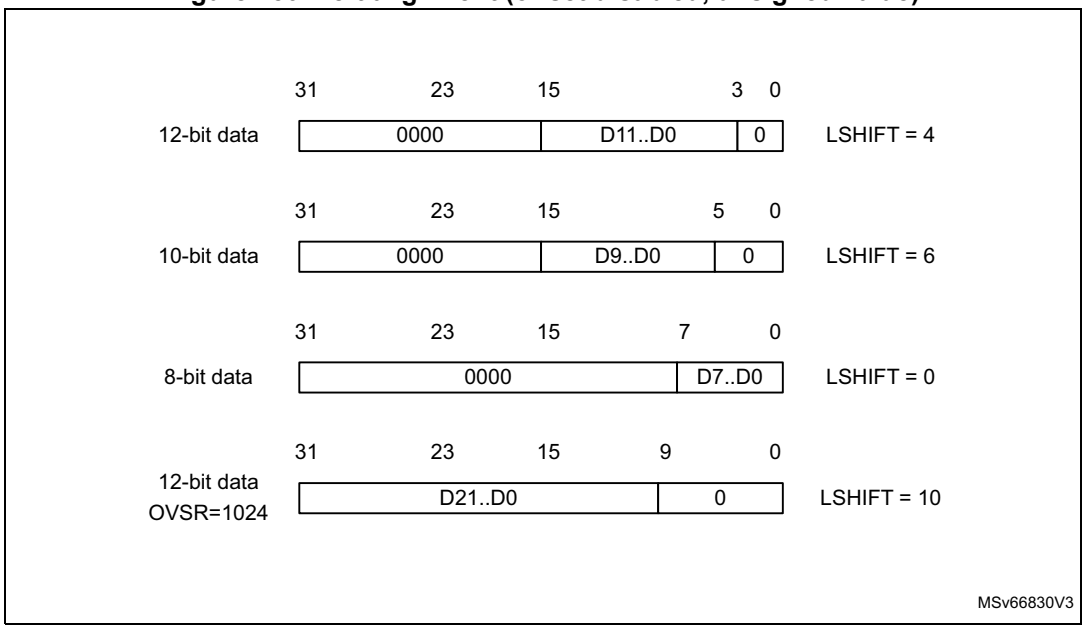
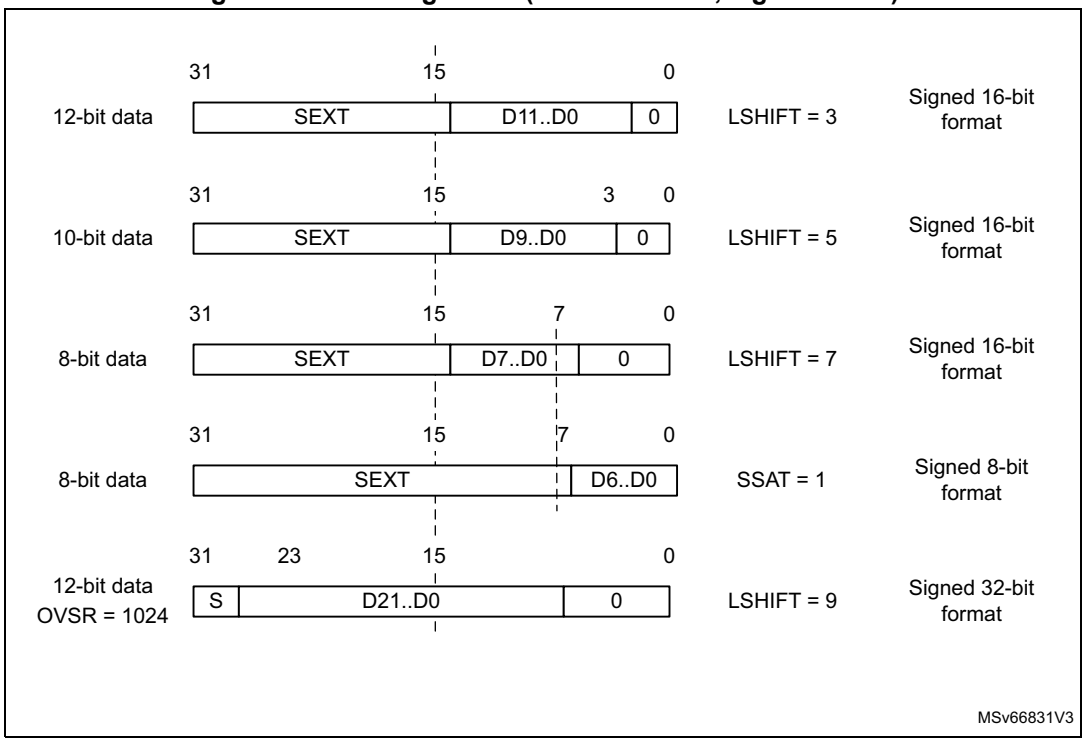


Figure 151. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 183 shows the sign-extended data format corresponding to different resolutions.

Table 183.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 183.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 184 provides numerical examples for four different offset values.

Table 184. Numerical examples for 32-bit or 16-bit format

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB−1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC conversion latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data.

The OVR flag is set when a new conversion completes while ADC_DR register FIFO was full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode nor ADF output mode cannot be supported.

Figure 152. Example of overrun (OVRMOD = 0)

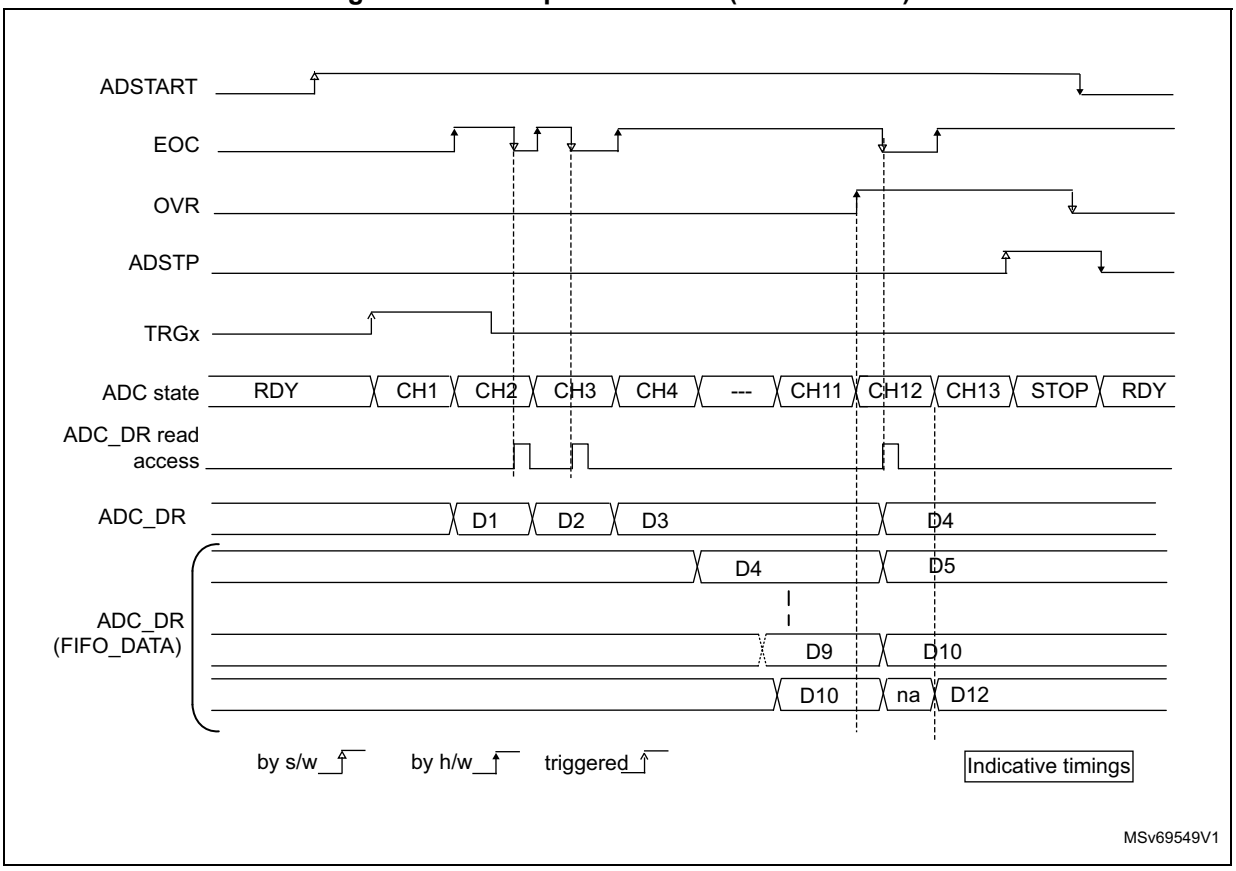
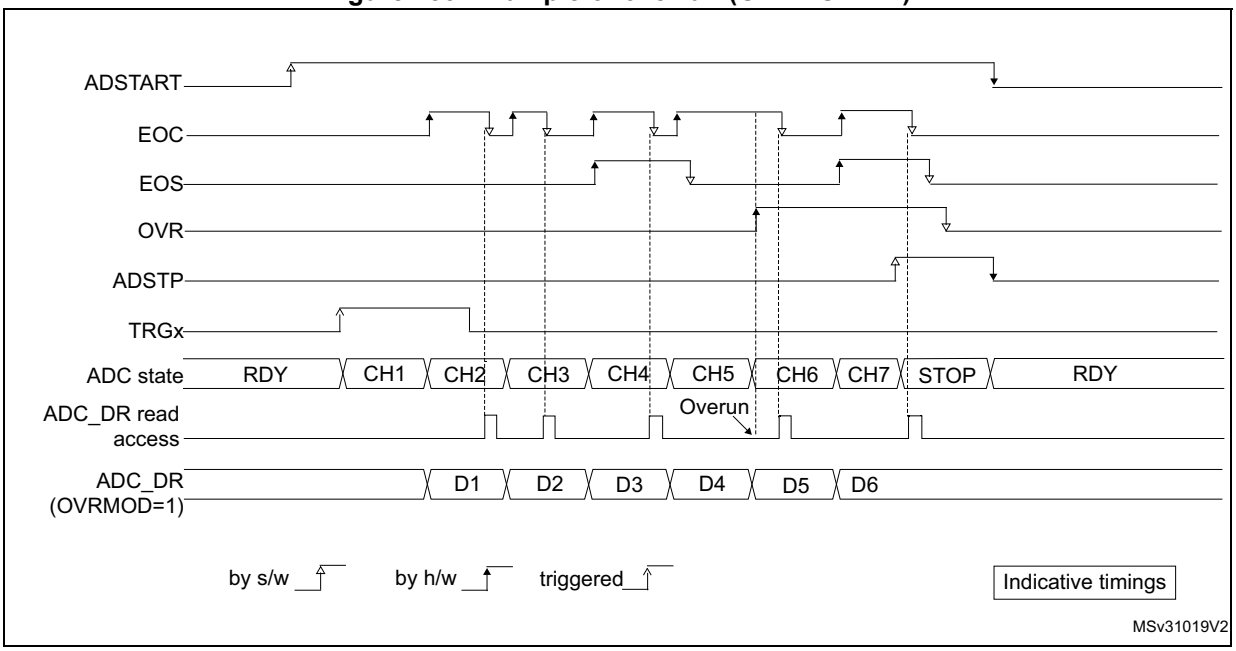


Figure 153. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode or DAMDF different from 00 in dual ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

Two different DMA modes are proposed depending on the application usage. They can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register in single ADC mode and the DAMDF bit of the ADCC12_CCR register in dual ADC mode, respectively:

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA in circular mode.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen,
- All ongoing conversions are aborted with the partial result discarded,
- No new DMA request is issued to the DMA controller in order to avoid generating an overrun error if there are still conversions that are started,
- The scan sequence is stopped and reset, and
- The DMA is stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA with FIFO

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

22.4.27 Managing conversions using the ADF

The ADC conversion results can be transferred directly to the ADF.

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers the 16 least significant bits of the regular data register to the ADF through adcx_dat[15:0] bus, which in turn resets the EOC flag once the transfer is effective.

The data format must be in 16-bit signed format:

ADC_DR[31:16] = Don’t care

ADC_DR[15] = Sign

ADC_DR[14:0] = Data

Any value above the 16-bit signed format is truncated.

To obtain a 16-bit signed format, the OFFSET function needs to be activated.

In this mode, OVRMOD = 1 is not supported.

22.4.28 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- All the previous data of the same group have been treated, and
- The ADC_DR register has been read or the EOC bit has been cleared (see *Figure 154*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 155*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored. This is not the case for software triggers for which the ADSTART and JADSTART bits can still be set during the autodelay to restart the conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 155*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 157*).

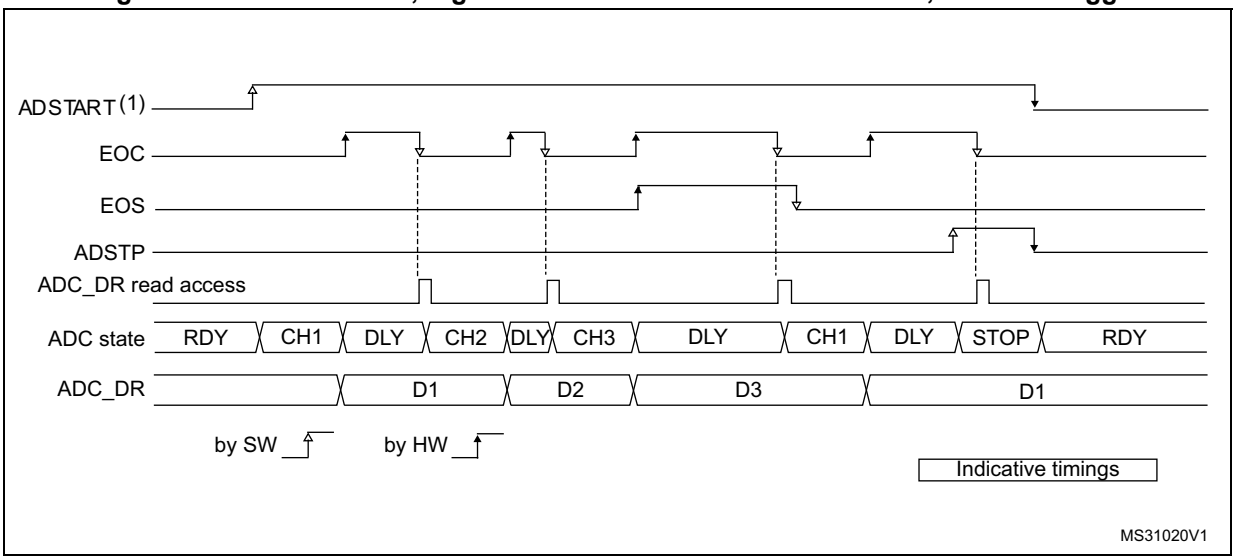
The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 158*).

Caution: In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

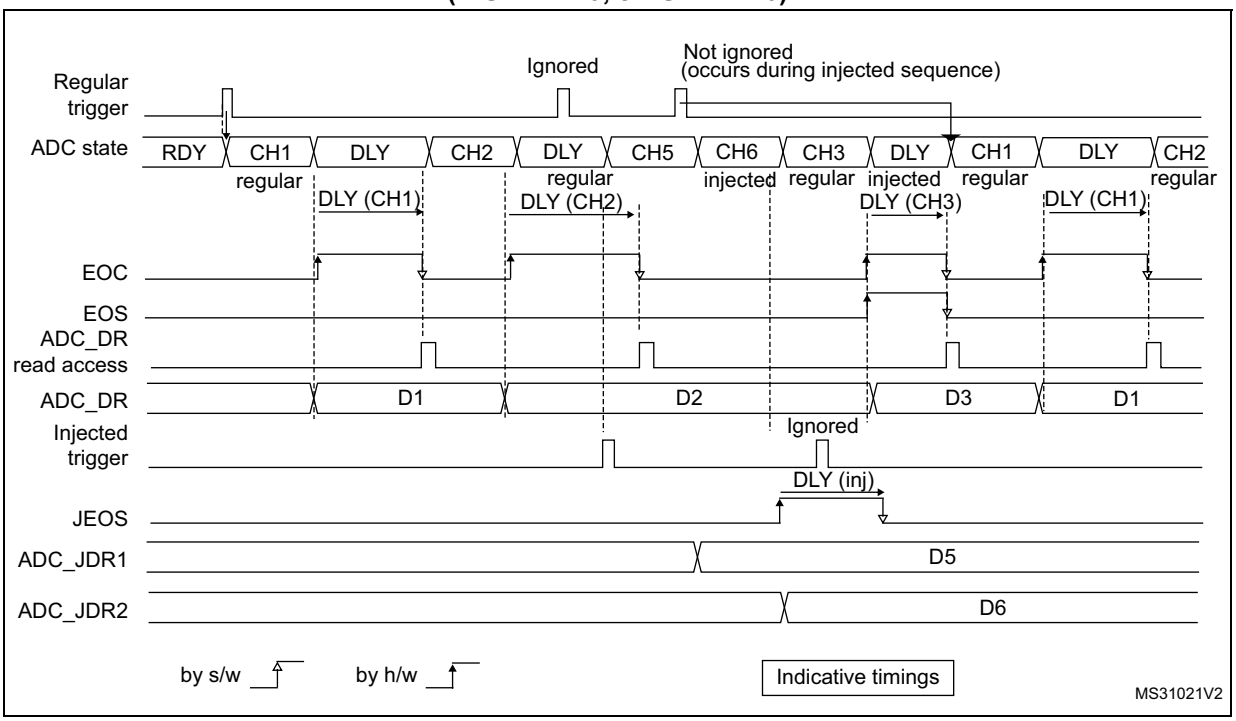
In autodelay mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 154. AUTODLY = 1, regular conversion in continuous mode, software trigger



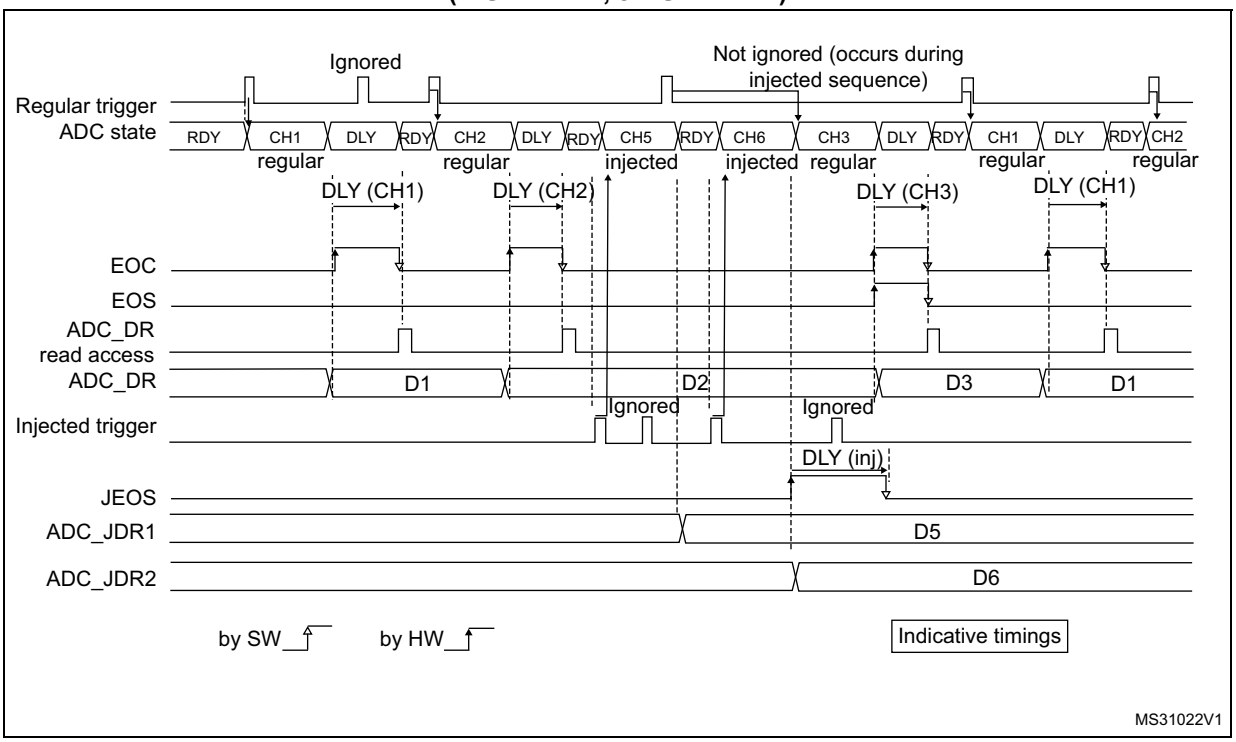
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1,2,3.
3. Injected configuration DISABLED.

Figure 155. AUTODLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



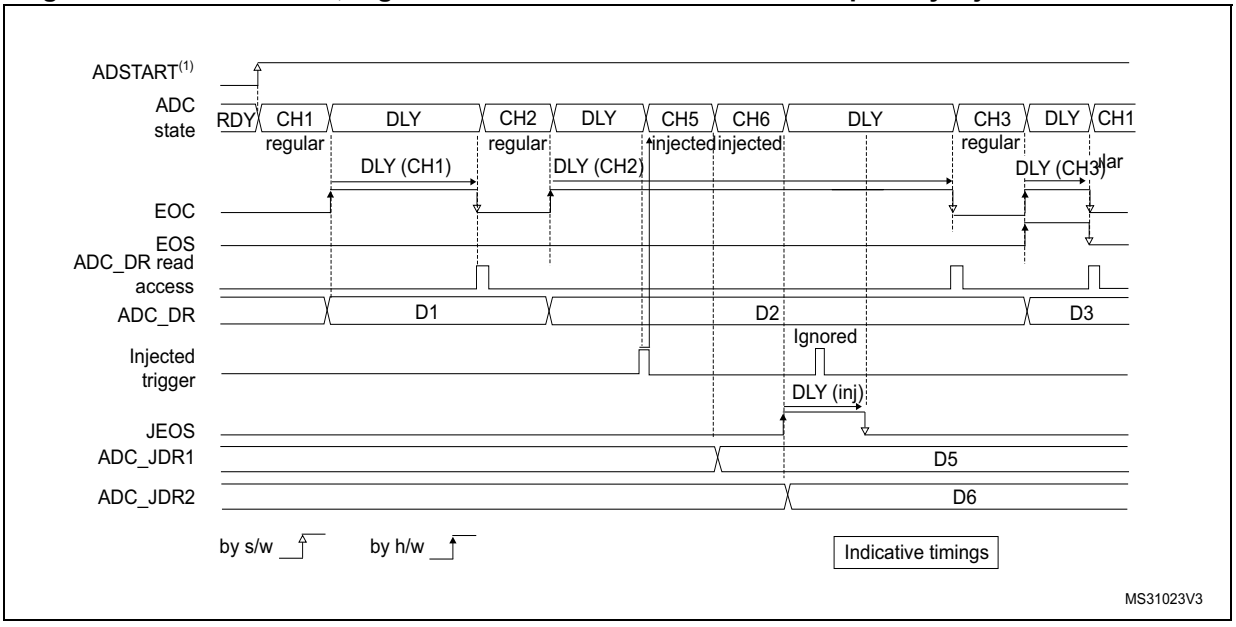
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5,6.

Figure 156. AUTODLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



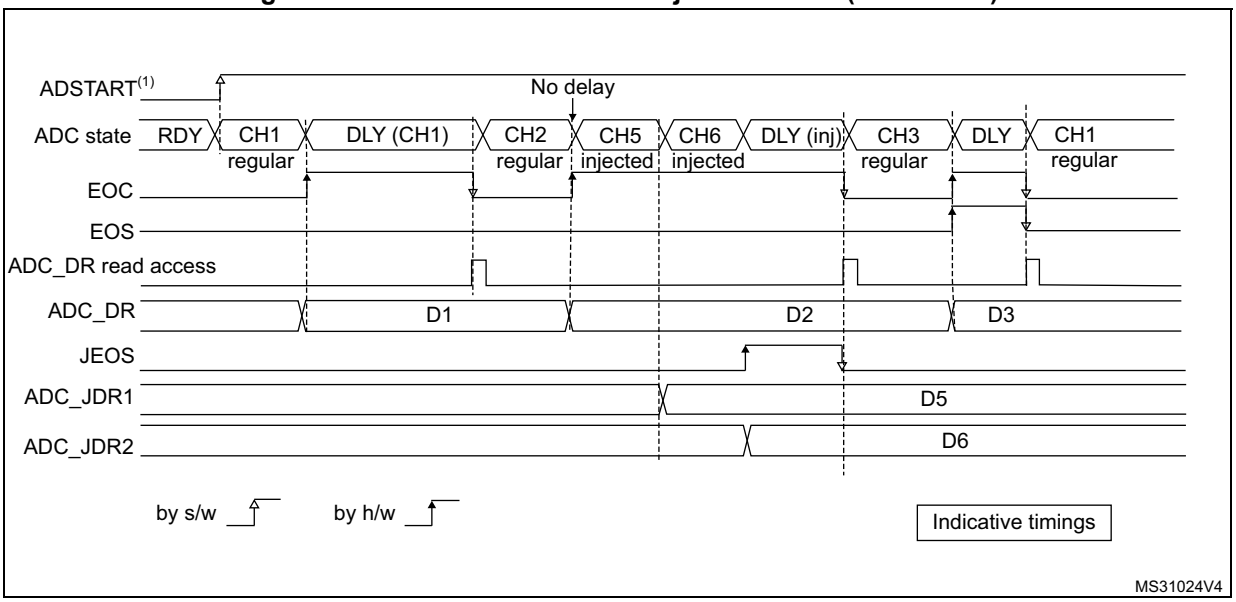
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5,6.

Figure 157. AUTODLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5,6.

Figure 158. AUTODLY = 1 in autoinjection mode (JAUTO = 1)

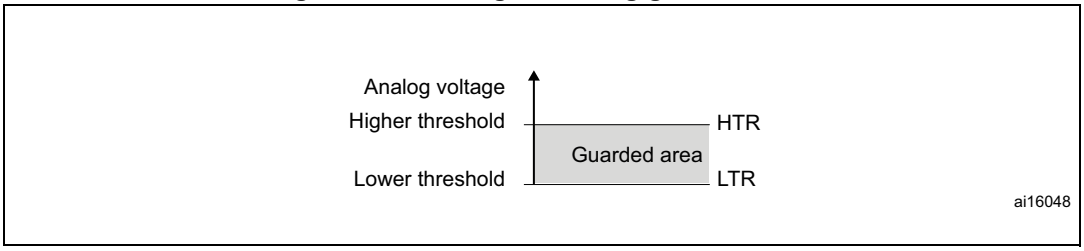


1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6.

22.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 159. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in scan mode is not allowed), a valid ADC conversion data range can be configured through the ADC_AWD1LTR and ADC_AWD1HTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWD1_OUT is generated as well as the AWD1 flag. Before this value is reached, AWD1 output is not activated.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 185 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 185. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 186 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 186. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[18:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[18:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 160. ADC_AWDy_OUT signal generation (on all regular channels)

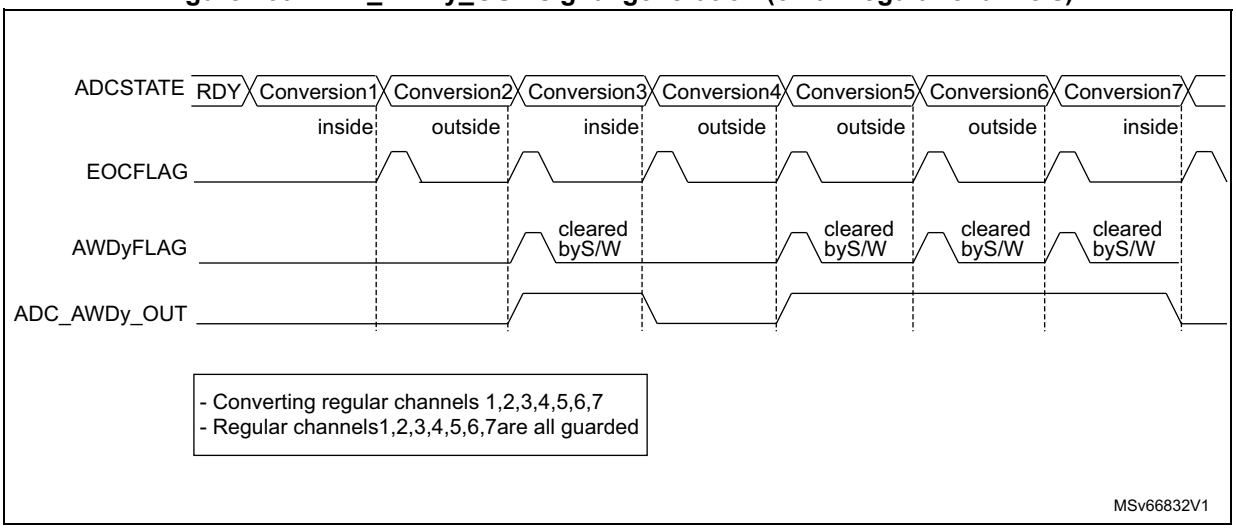


Figure 161. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

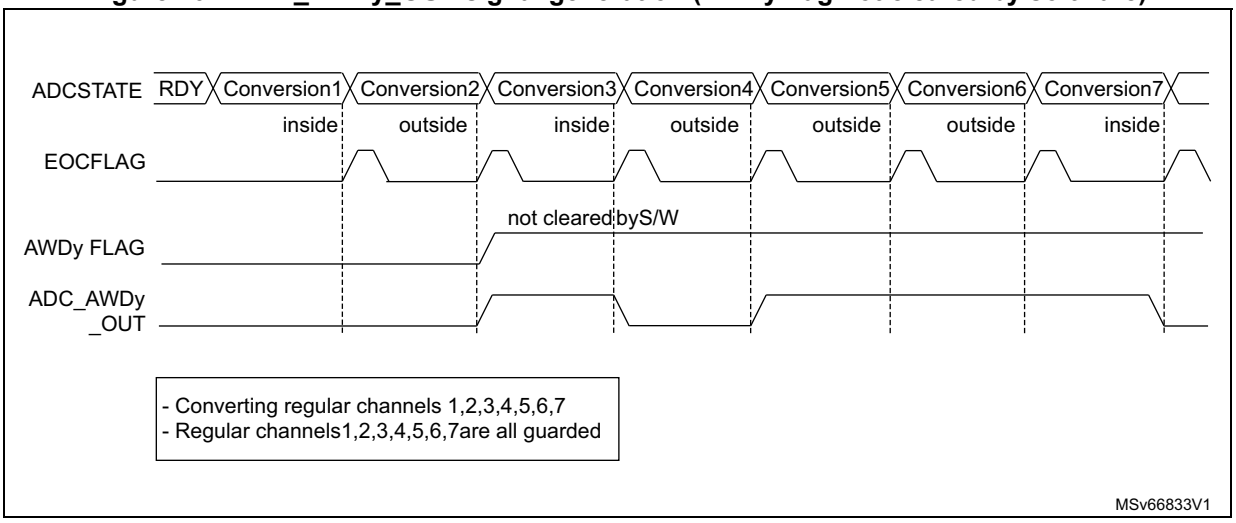


Figure 162. ADC_AWDy_OUT signal generation (on a single regular channel)

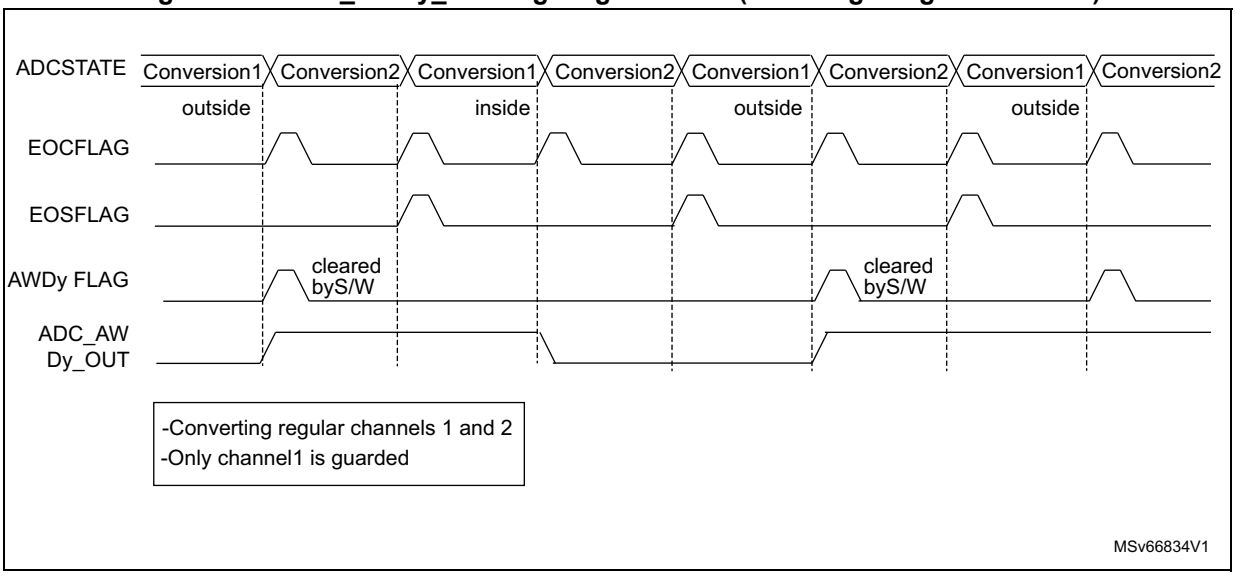
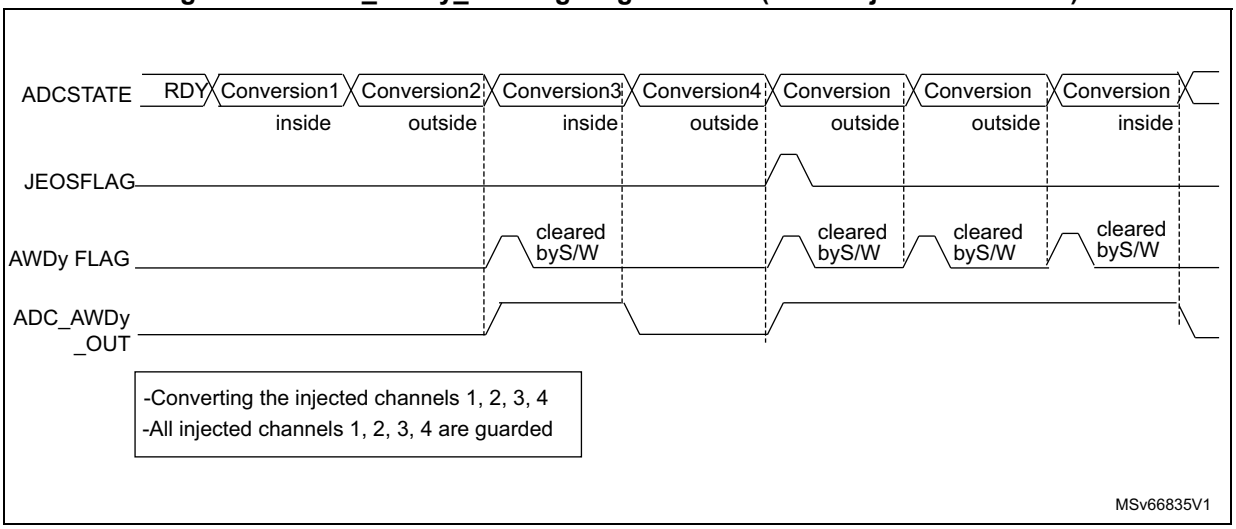


Figure 163. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

22.4.30 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined through the OVSR[9:0] bits of the ADC_CFGR2 register. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register.

The summation unit can yield a result up to 22 bits, which can be left or right shifted. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

Figure 164 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 164. 12-bit result oversampling with 10-bit right shift and rounding

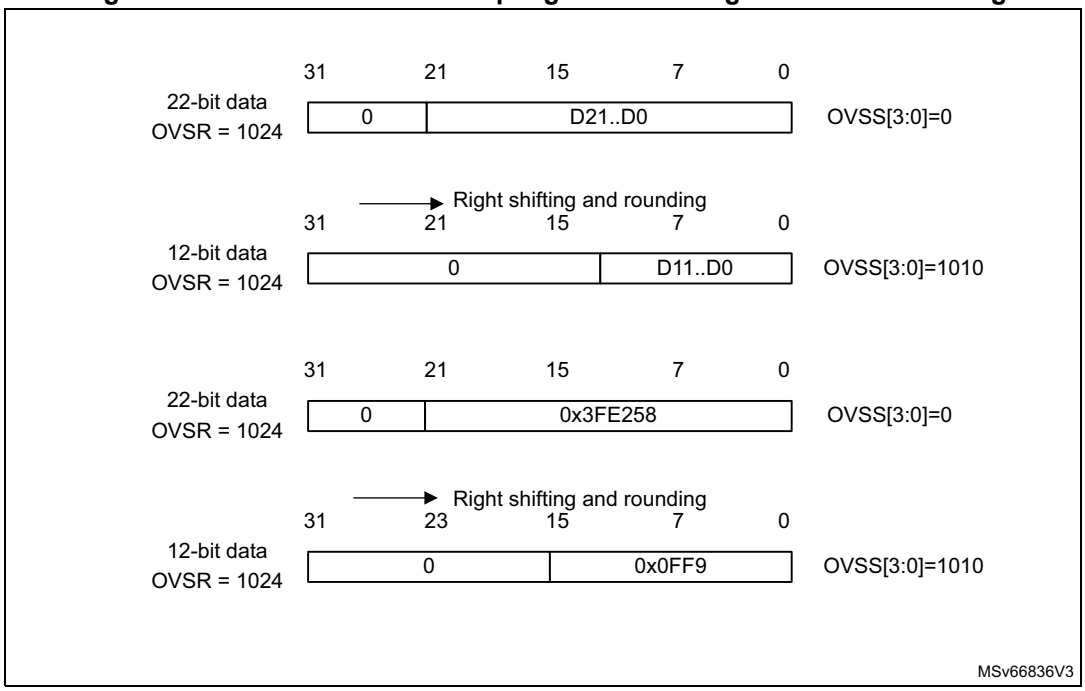


Table 152 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 187. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the conversion latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and right-shift. The EOC flag is set two additional clock cycles after the conversion is complete.

ADC operating modes supported in oversampling mode (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous conversion modes
- ADC conversions started either by software or triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits of the ADC_CFGR1 register) are accumulated, truncated, rounded, and shifted in the same way as 12-bit conversions are.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

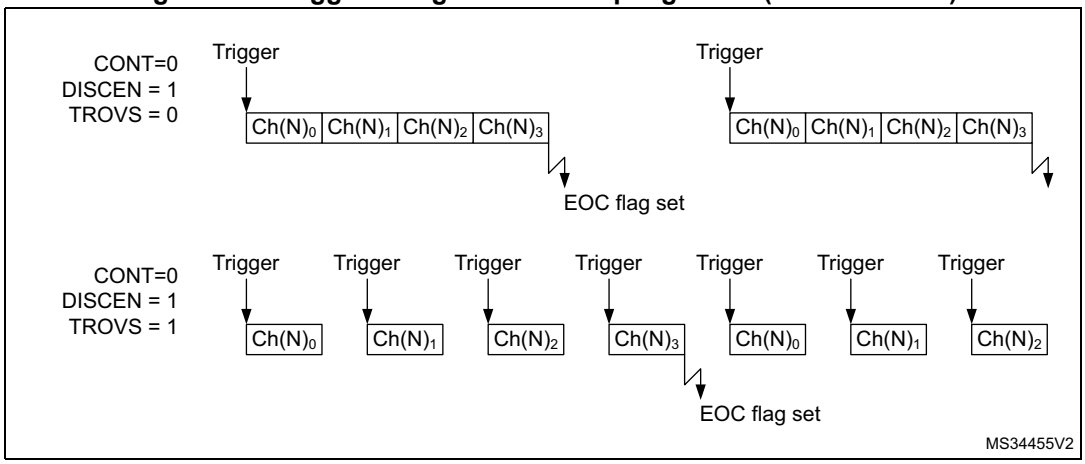
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 165 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither JAUTO nor AUTODLY is supported.

Figure 165. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

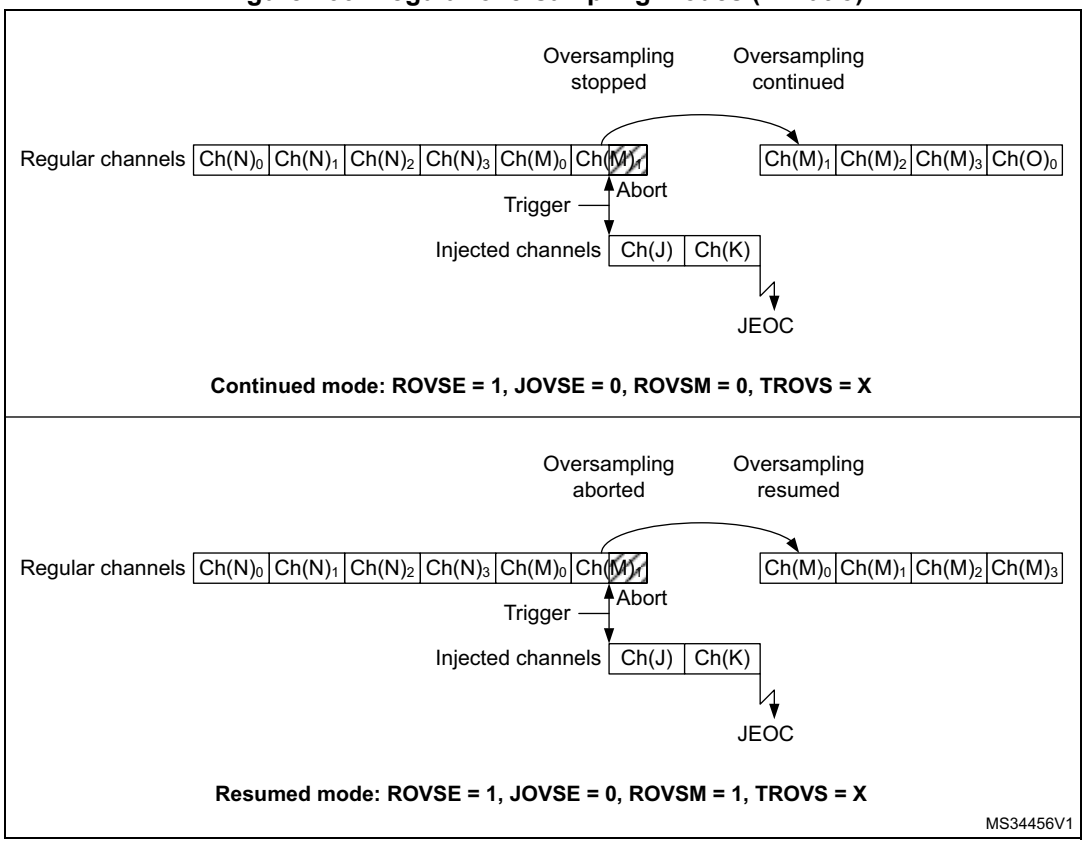
The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is

complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);

- In resumed mode, the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequence is blocked.
- *Figure 166* gives examples for a 4x oversampling ratio.

Figure 166. Regular oversampling modes (4x ratio)



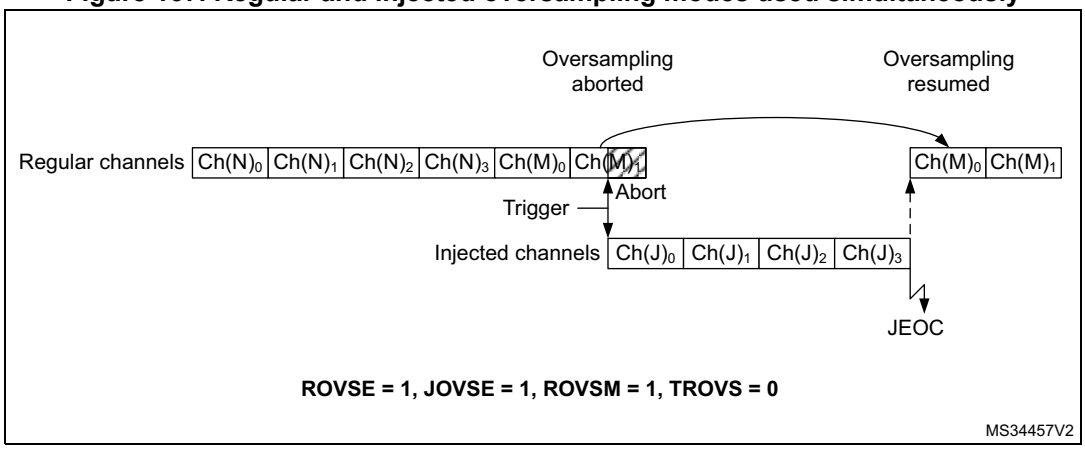
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 167*.

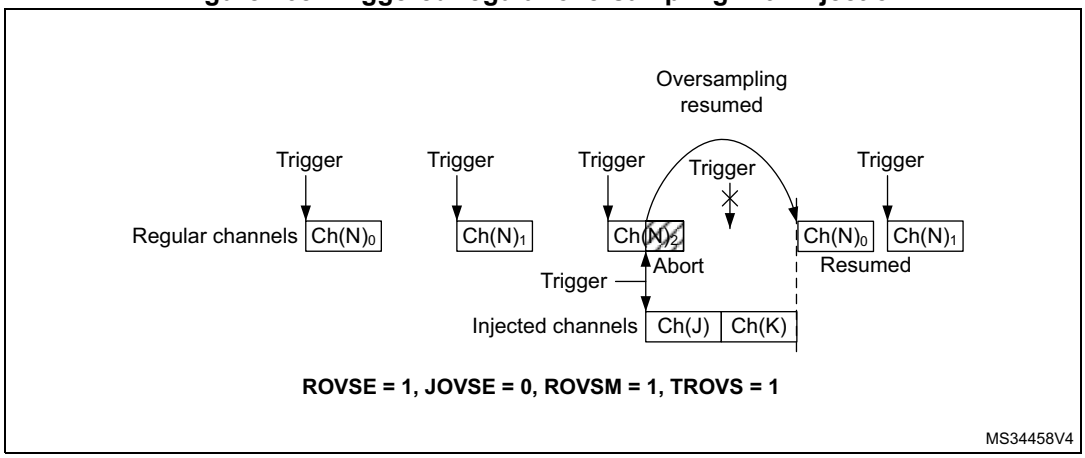
Figure 167. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled and the ROVSM bit set (Resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 168*.

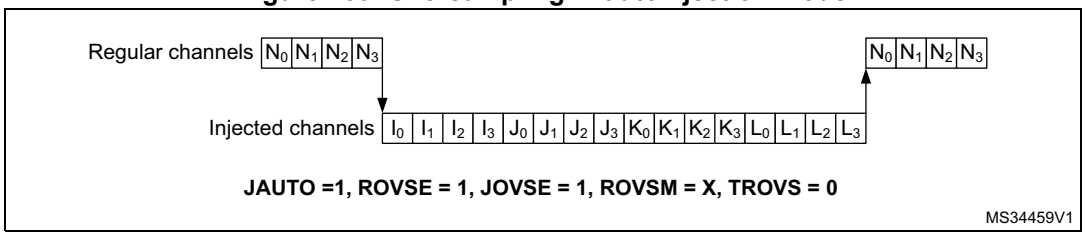
Figure 168. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 169* shows how the conversions are sequenced.

Figure 169. Oversampling in autoinjection mode



In addition, the triggered mode can be enabled using the TROVS bit. In this case, the ADC must be configured as follows: JAUTO = 1, DISCEN = 0, JDISCEN = 0, DISCNUM = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Combined modes summary

Table 188 summarizes all combinations, including the modes that are not supported.

Table 188. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

22.4.31 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 170*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when DUAL[4:0] bits in the ADCC_CCR register are not equal to zero), the CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JAUTO bits of the ADC_CFGR1 register are shared between the master and slave ADC: the slave ADC bits are ignored and always apply to the corresponding bits of the master ADC.

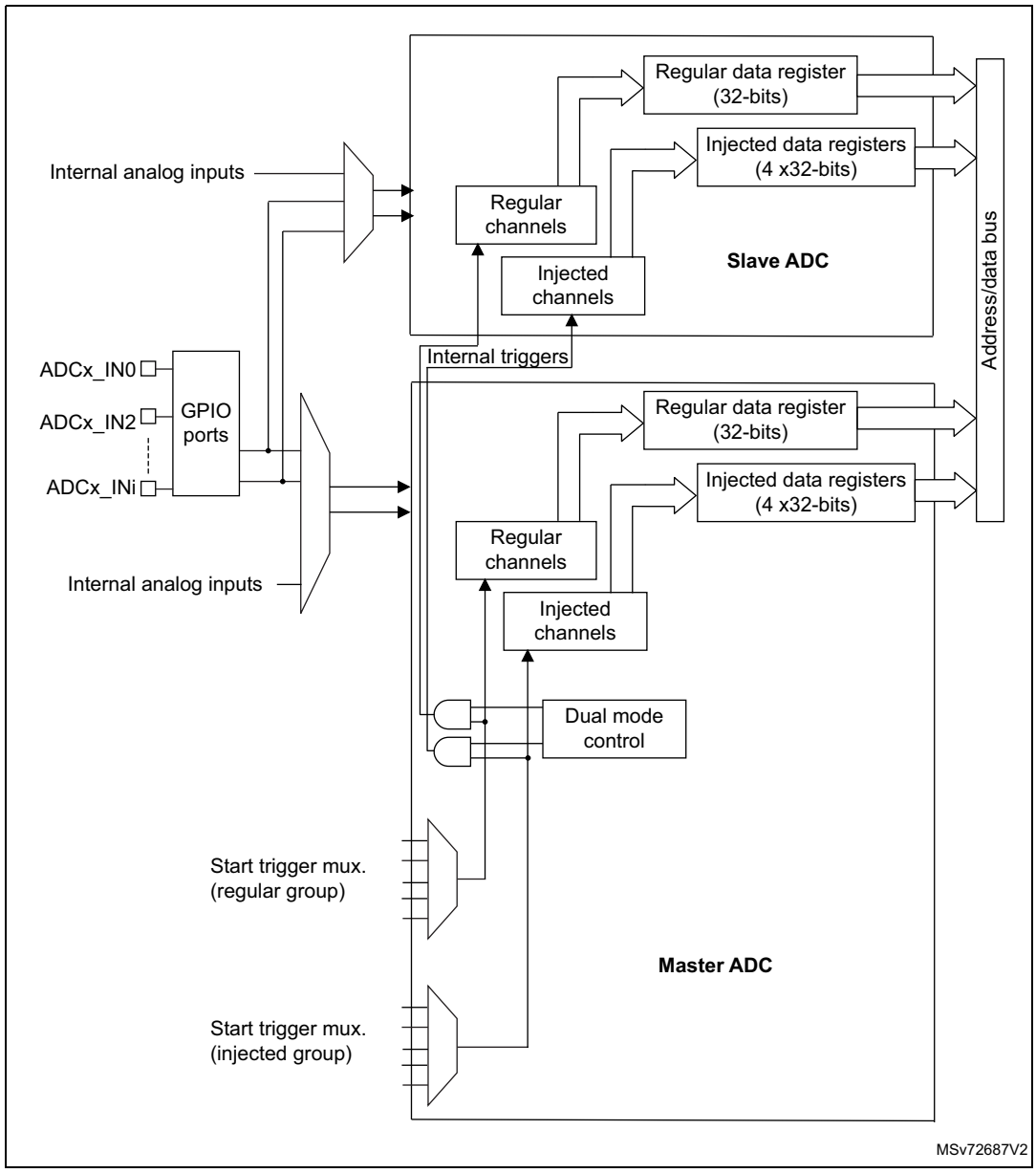
To start a conversion in dual mode, the user must program the EXTEN[1:0], EXTSEL, JEXTEN[1:0] and JEXTSEL bits of the master ADC only to configure a software or hardware trigger and a regular or injected trigger (the EXTEN[1:0] and JEXTEN[1:0] bits of the slave ADC are don't care).

In regular simultaneous or interleaved modes, once the user sets the ADSTART or ADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, ADSTART or ADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes, once the user sets the JADSTART or JADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, JADSTART or JADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCC_CDR). The flags can also be read in parallel by reading the dual-mode status register (ADCC_CSR).

Figure 170. Dual ADC block diagram



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

The injected simultaneous mode is selected by programming DUAL[4:0] bits to 0b00101.

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

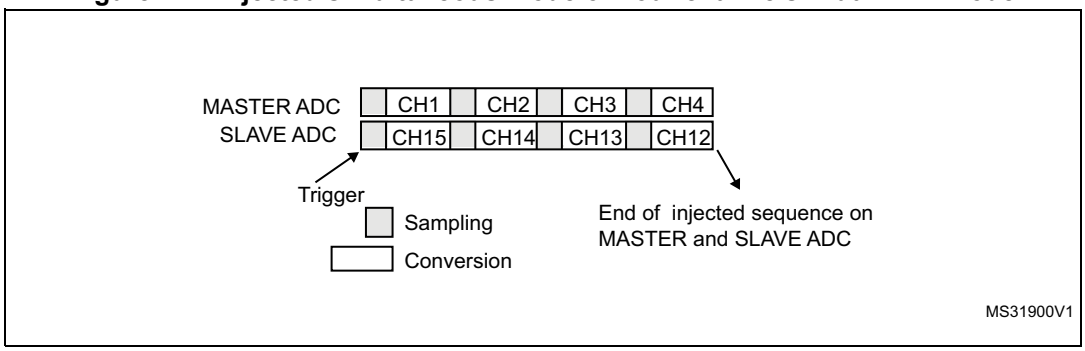
In simultaneous mode, the user software must convert sequences with the same length.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

The software is notified by interrupts when it can read the data:

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master-injected sequence is equal to the duration of the slave injected one (as shown in *Figure 171*), the software can enable only one of the two JEOS interrupts (for example the master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 171. Injected simultaneous mode on four channels: Dual ADC mode



If JDISCEN is set, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with the AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected conversions

The regular simultaneous mode is selected by programming DUAL[4:0] bits to 0b00110.

This mode applies to a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR1 register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on the master or on the slave) aborts the current simultaneous conversions both for master and slave. They are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, the user software must convert sequences with the same length.

The software is notified by interrupts when it can read the data:

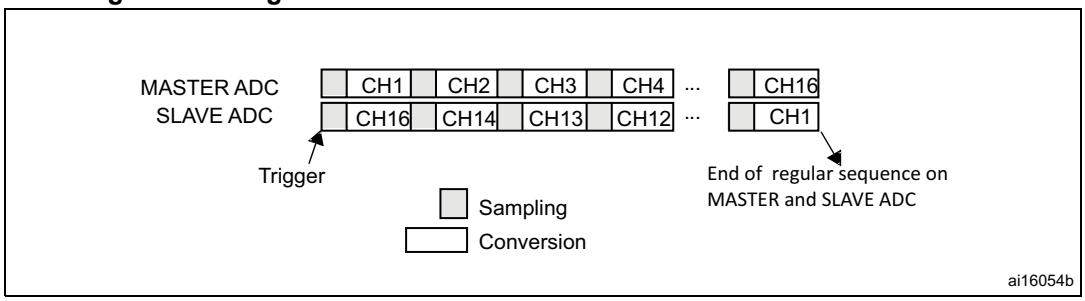
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (as shown in *Figure 172*), the software can enable only one of the two EOC interrupts (for example the master EOC) and read both converted data from the common data register (ADCC_CDR).

The regular data can also be read using the DMA. Several methods are possible:

- Use two DMA channels (one for the master and one for the slave). In this case DAMDF[1:0] bits must be kept cleared:
- Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
- Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configure the dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  a) Set DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
  b) A single DMA channel is used (the one corresponding to the master). Configure the DMA master ADC channel to read the common ADC register (ADCC_CDR).
  c) A single DMA request is generated each time both master and slave EOC events occur. At that time, the slave ADC converted data are available in the upper half-word of the ADCC_CDR 32-bit register and the master ADC converted data available in the lower half-word of the ADCC_CDR register.
  d) Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Note: When DAMDF[1:0] = 0b10 or 0b11, the user must program the same number of conversions in the master and in the slave sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 172. Regular simultaneous mode on 16 channels: dual ADC mode



If the DISCEN bit is set, each "*n*" simultaneous conversion of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with the AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence starts only if the common data register (ADCC_CDR) (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADCC_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

The DMA can be used to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master sequence is equal to the number of conversions in the slave sequence.
- For each simultaneous conversion of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert, the sampling time and the resolution of each channel.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected conversions

This mode is selected by programming DUAL[4:0] bits to 0b00111.

It can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately,
- The slave ADC starts after a delay of several ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay between two conversions in interleaved mode is configured in the DELAY bits in the ADCC_CCR register (see *Table DELAY bits versus ADC resolution*). This delay starts counting one half cycle after the end of the sampling phase of the master

conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However, the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: In case of 16-bit data format, only the EOC interrupt of the slave can be enabled and the common data register (ADCC_CDR) can be read. For 32-bit data format, enable both the slave and master EOC interrupts and the ADCC_CDR2 register. However, in this case, the user must make sure that the durations of the conversions are compatible to ensure that, inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the DAMDF mode.

Regular data can also be transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the DAMDF mode, as described below:

1. Configure DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
2. A single DMA channel is used (the one of the masters). Configure the DMA master ADC channel to read the common ADC register (ADCC_CDR).
3. A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADCC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of the ADCC_CDR register.
4. Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Table 189. DELAY bits versus ADC resolution(1)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

Table 189. DELAY bits versus ADC resolution(1) (continued)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
1010	11 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1100	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
others	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

1. Preliminary information, subject to change.

Figure 173. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode

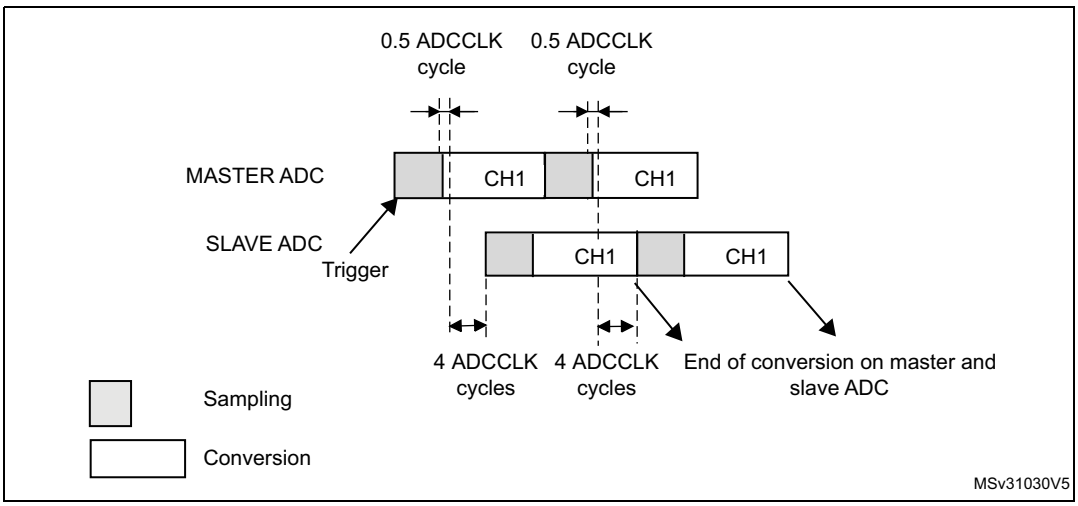
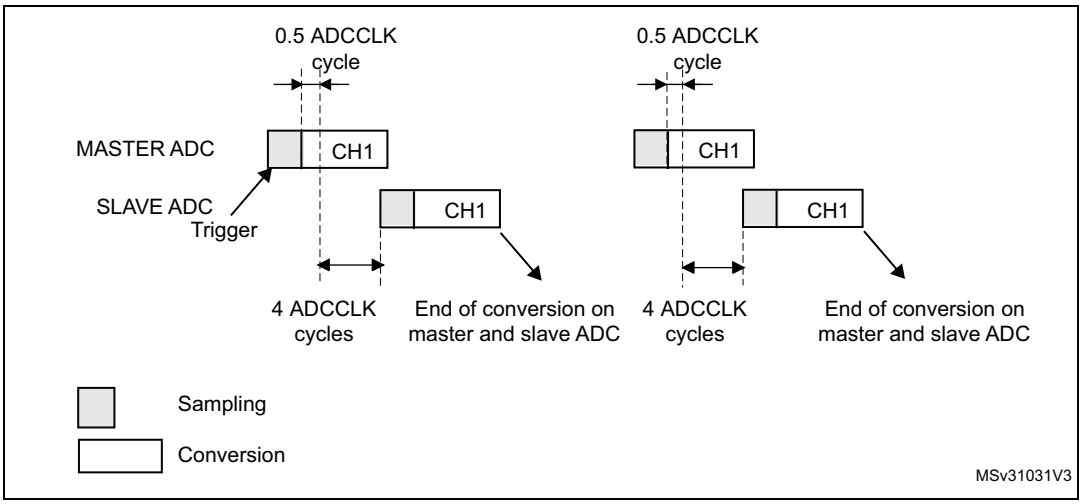


Figure 174. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode



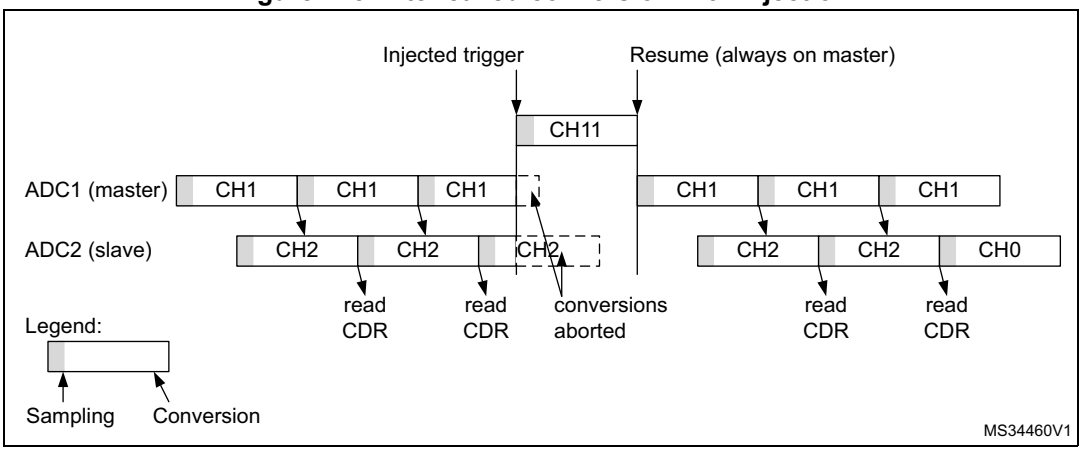
If the DISCEN bit is set, each "*n*" simultaneous conversion ("*n*" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In interleaved mode, injected conversions are supported. When injection is done (either on master or slave), both master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 175*).

Note: In interleaved mode, the ADCx_AWDy_OUT signal for the master ADC is synchronized with the EOC of the master ADC, but the interrupt flag is synchronized with the EOC/JEOC flag of the slave ADC.

In addition, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 175. Interleaved conversion with injection



Alternate trigger mode

The alternate trigger mode is selected by programming DUAL[4:0] bits to 0b01001.

This mode can only be started on an injected group. The source of the external trigger comes from the multiplexer injected group of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 0x0.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the first trigger occurs, all injected master ADC channels in the group are converted.
2. When the second trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on...

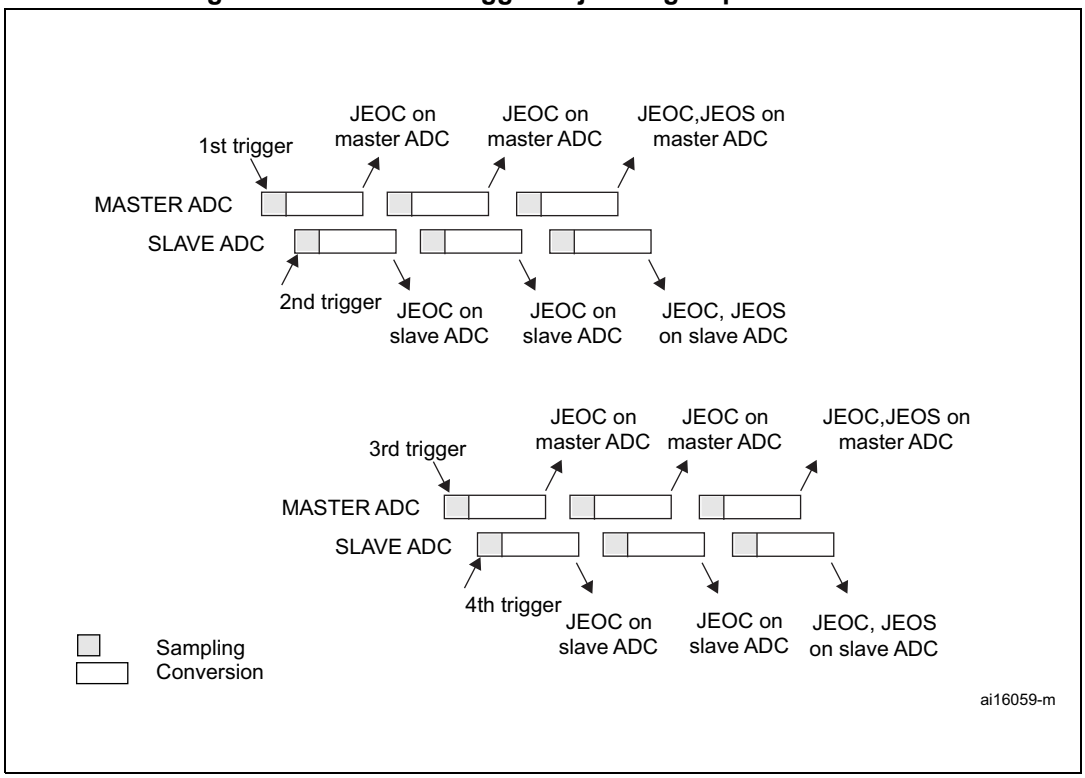
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 176. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case, the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between two trigger events must be greater than or equal to one ADC clock period. The minimum time interval between two trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

1. When the first trigger occurs, the first injected channel of the master ADC is converted.
2. When the second trigger occurs, the first injected channel of the slave ADC is converted.
3. And so on...

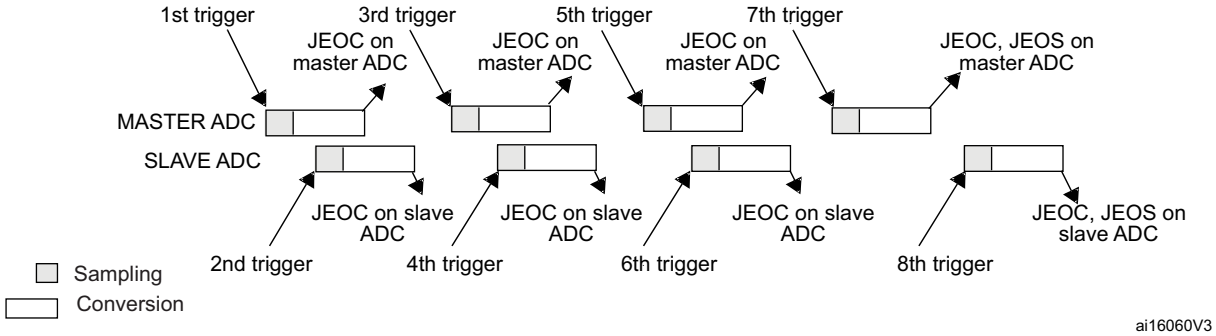
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted, then the alternate trigger process restarts.

Figure 177. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming DUAL[4:0] bits to 0b00001.

The simultaneous conversion of a regular group can be interrupted to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, the user software must convert sequences with the same length or ensure that the interval between triggers is longer than the sum of the conversion times of the two sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous and alternate trigger mode

This mode is selected by programming DUAL[4:0] bits to 0b00010.

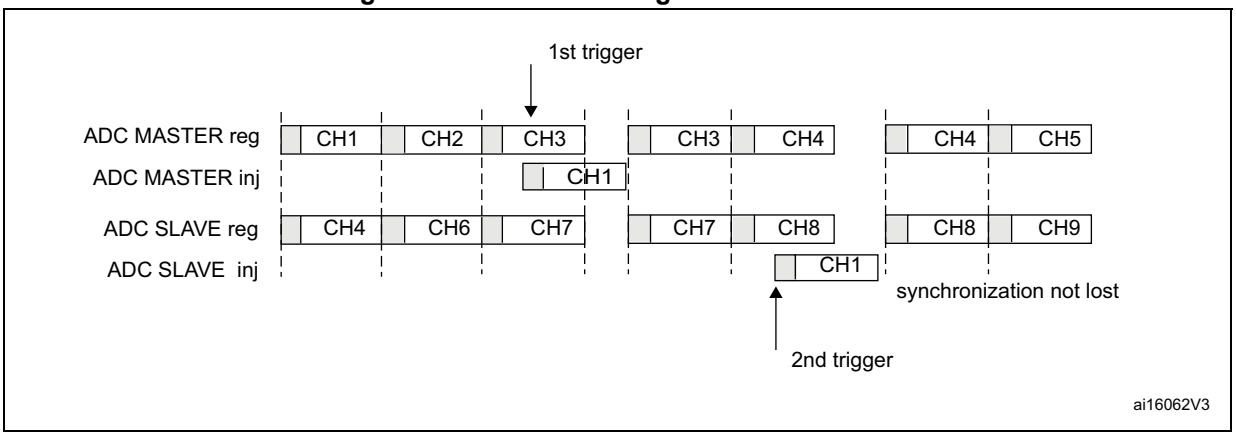
The simultaneous conversion of a regular group can be interrupted to start the alternate trigger conversion of an injected group. *Figure 178* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion immediately starts after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, the user software must convert sequences with the same length or ensure that the interval between triggers is longer than the sum of the conversion times of the two sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

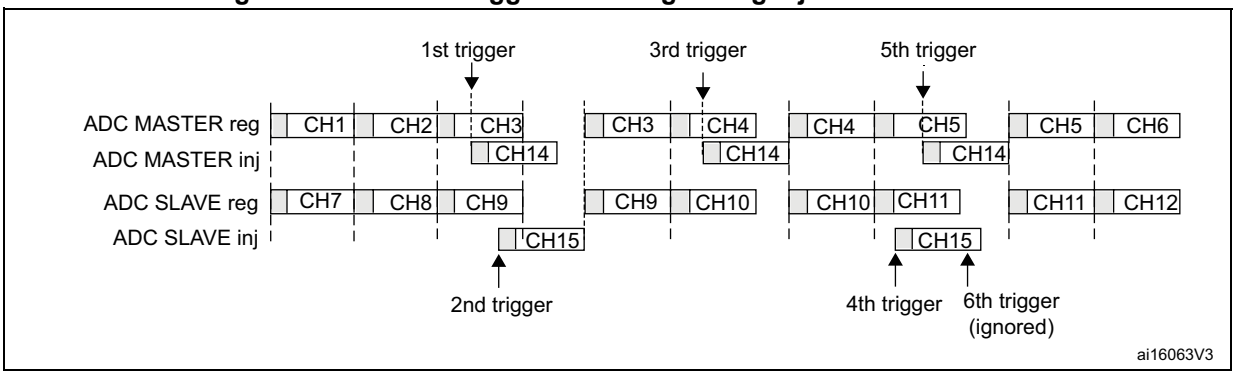
The software trigger is not supported for injected conversions.

Figure 178. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 179* shows the behavior in this case (note that the sixth trigger is ignored because the associated alternate conversion is not complete).

Figure 179. Case of trigger occurring during injected conversion



Combined injected simultaneous and interleaved mode

This mode is selected by programming DUAL[4:0] bits to 0b00011.

An interleaved conversion can be interrupted with a simultaneous injected event.

In this case, the interleaved conversion is immediately interrupted and the simultaneous injected conversion starts. At the end of the injected sequence, the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 180*, *Figure 181* and *Figure 182* show examples of this behavior.

Note: In this mode, the ROVSM =0 setting is not supported in oversampling mode.

Caution: In this mode, it is mandatory to use the common data register to read the regular data with a single read access. Otherwise, master-slave data coherency is not guaranteed.

Figure 180. Interleaved single channel CH0 with injected sequence CH11, CH12

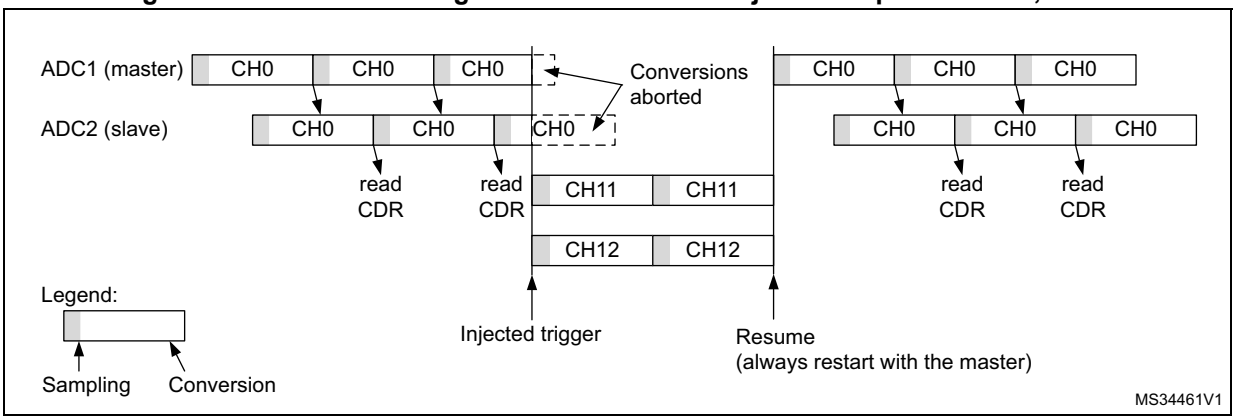


Figure 181. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

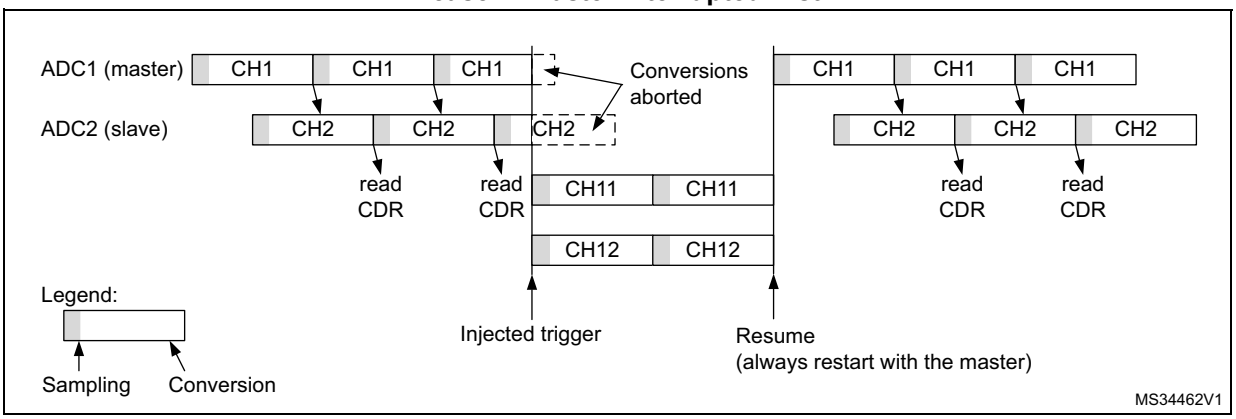
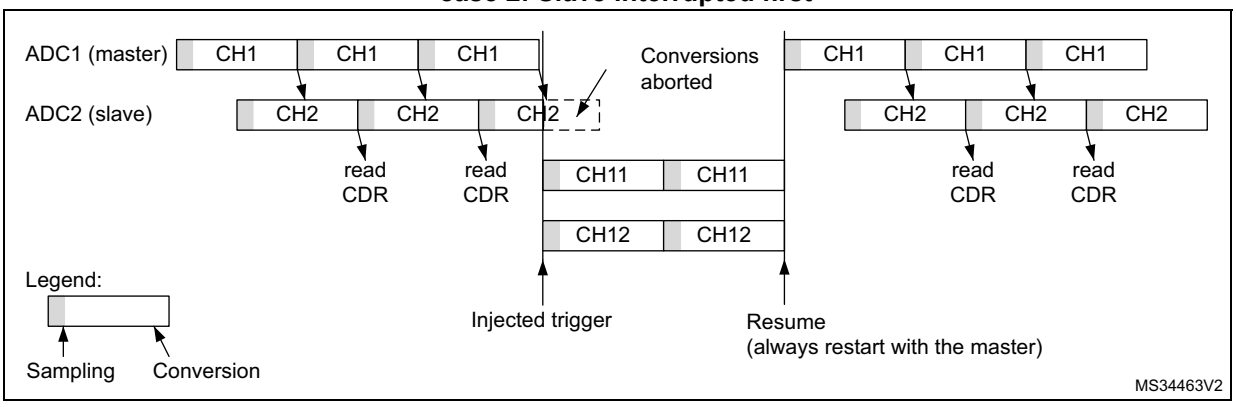


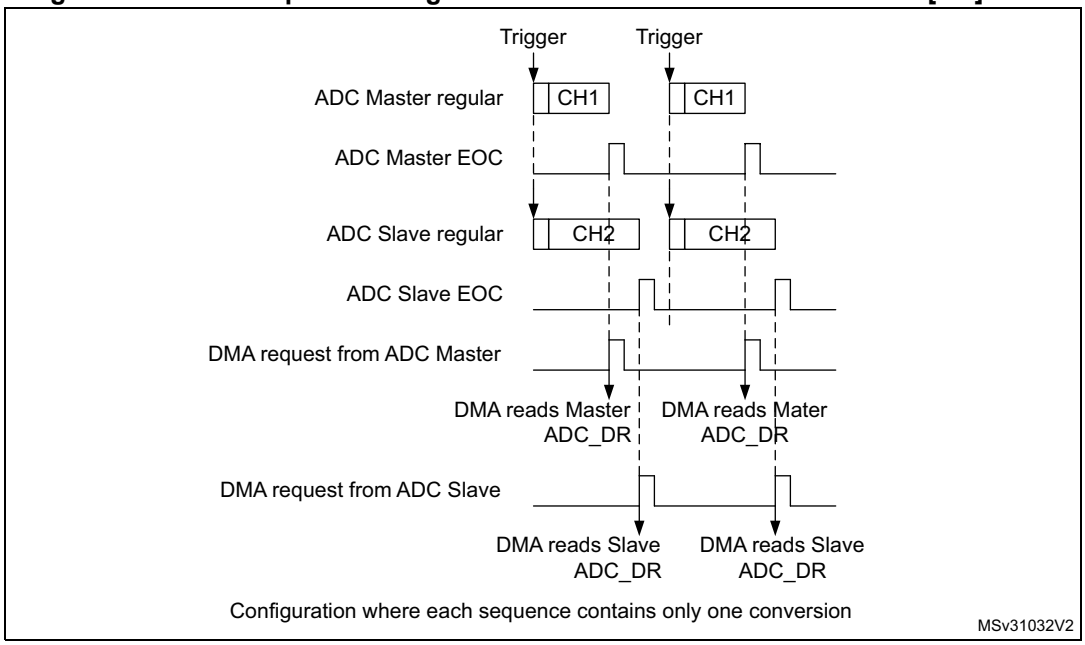
Figure 182. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 183: DMA Requests in regular simultaneous mode when DAMDF[1:0]=0b00*).

Figure 183. DMA Requests in regular simultaneous mode when DAMDF[1:0]=0b00



In interleaved mode, it is also possible to save one DMA channel and transfer both data using a single DMA channel. To do this, the DAMDF[1:0] bits must be configured in the ADCC_CCR register:

- **DAMDF[1:0] = 0b10, 32-bit format:**

  A single DMA request is generated alternatively after either the master or slave EOC events have occurred. At that time, the data are alternatively available in the 32-bit register ADCC_CDR2 32-bit register.

  This mode is used in interleaved mode and in regular simultaneous mode when the data width is above 16 bits (conversion data width can exceed 16 bits when functions such as oversampling, compensation and left-bit shift, are used).

  Example:

  Interleaved dual mode: a DMA request is generated each time a new 32-bit data is available:

  First DMA request: ADCC_CDR2[31:0] = MST_ADC_DR[31:0]

  Second DMA request: ADCC_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF[1:0] = 0b10, 16-bit format**:

  A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCC_CDR contains the two half-words representing two ADC-converted data. The slave ADC data takes the upper half-word and the master ADC data takes the lower half-word.

  This mode is used in interleaved mode and in regular simultaneous mode when the data width is less than 16 bits. Any value above 16-bit in the master or the slave-converted data is truncated to the least 16 significant bits.

Example:

Interleaved dual mode: a DMA request is generated each time a new 32-bit data is available:

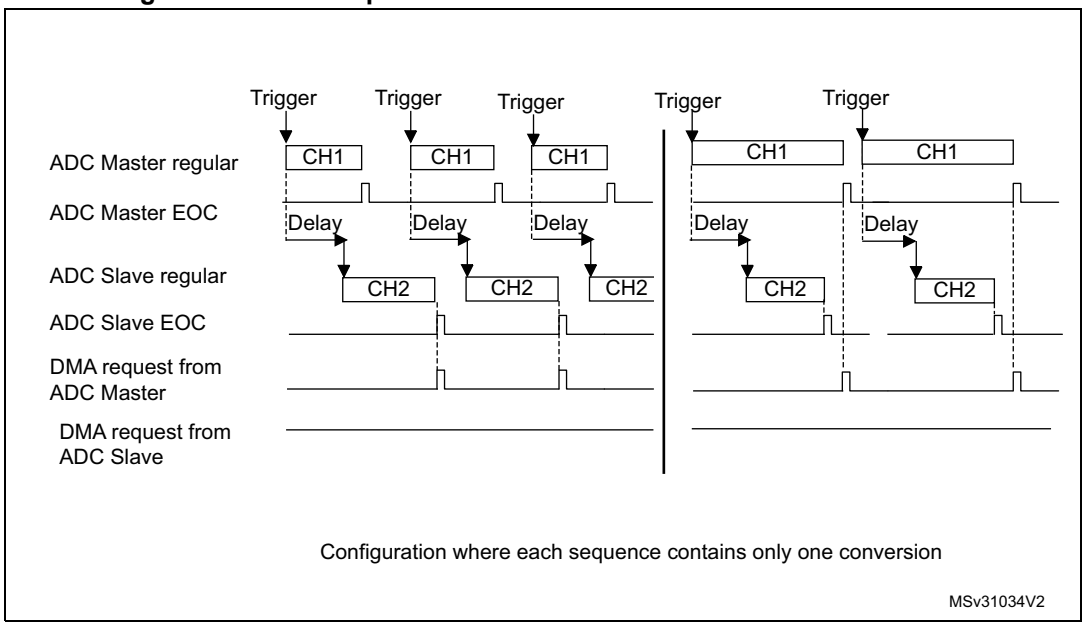
First DMA request:

ADCC_CDR[31:0] = (SLV_ADC_DR[15:0] << 16) | MST_ADC_DR[15:0]

Second DMA request:

ADCC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 184. DMA requests in interleaved mode when DAMDF = 0b10



Note: When using dual ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion data is available.

- **DAMDF[1:0] = 0b11**:

  This mode is similar to the DAMDF[1:0] = 0b10 configuration. The only difference is that on each DMA request (two data are available), two bytes representing two ADC converted data are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when the resolution is 6 bits or 8 bits and data are not signed

  Example:

  Interleaved dual mode: a DMA request is generated each time two data items are available:

  First DMA request:

  ADCC_CDR[15:0] = (SLV_ADC_DR[7:0] << 8)| MST_ADC_DR[7:0]

  Second DMA request:

  ADCC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Note: DMA burst mode is not supported in ADC dual mode.

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). The EOC bit corresponding to one ADC might remains set because the data register of this ADC contains valid data.

DMA one-shot mode/ DMA circular mode when multi-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), DMNGT[1:0] bits of the ADCC_CCR register must also be configured to select between DMA one-shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*

Stopping the conversions in dual ADC modes

In dual ADC mode, the user software must set the ADSTP/JADSTP control bits in the master ADC to stop the conversions of both ADC. The other ADSTP control bit on the slave ADC has no effect in dual ADC mode. Except in DMA one-shot mode, the ADSTP control bit for the master ADC does not affect the slave ADC DMA request signal. In this case, the ADSTP bit must be set on the slave ADC.

Once both ADCs are effectively stopped, the ADSTART/JADSTART bits on the master and slave ADCs are both deasserted by hardware.

Interleaved mode with oversampling

When oversampling is enabled, the ROVSM bit must be set.

Regular sampling in continued mode is supported.

Analog watchdog in dual mode

The analog watchdog is supported in dual mode.

Disabling the ADC in dual mode

During a dual-mode operation, the master ADC shares the ADC clock with the slave ADC. To disable the ADC correctly through the ADDIS bit of ADC_CR, set the ADSTP (or JADSP) bit on the master ADC and, if DUAL[4:0] equals 0b00101, 0b00110, 0b00111, or 0b01001, set ADSTP (or JADSP) on the slave ADC as well. Once ADSTP (or JADSTP) becomes 0 both for the master and slave ADCs, set the ADDIS bit on the slave ADC, then set the ADDIS bit on the master ADC.

ADF mode in dual ADC interleaved mode

In dual ADC interleaved modes (DUAL[4:0] = 0b00011 or DUAL[4:0] = 0b00111), the ADC conversion results can be transferred directly to the ADF.

This mode is enabled by setting DMNGT[1:0] bits to 0b10 in the master ADC ADC_CFGR1 register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the ADF. Each transfer resets the EOC flag of each channel once the transfer is complete.

The data format must be 16-bit signed:

ADC_DR[15:12] = Sign extended

ADC_DR[11] = Sign

ADC_DR[10:0] = Data

To obtain 16-bit signed format, the software must configure OFFSET[11:0] bits to 0x800 with OFFSETy_CH[4:0] selecting the target channel set.

Only right aligned data can be set as ADF input format (see *Figure 149: Right alignment (offset enabled, signed value)*).

ADF mode in dual ADC simultaneous mode

It is not mandatory to use the ADF in dual ADC simultaneous mode since conversion data are handled by each individual channel. Single mode with same trigger source results in simultaneous conversion with ADF interface.

Dual ADC modes supported in oversampling mode

It is possible to have oversampling enabled when working in dual ADC configuration for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

22.4.32 Temperature sensor

The temperature sensor can be used to measure the junction temperature (T_J) of the device.

The temperature sensor is internally connected to the ADC input channels, which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 22.4.2: ADC pins and internal signals* for more details). When not in use, the sensor

can be put in power-down mode. It supports the temperature range specified in the product datasheet.

Figure 185 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

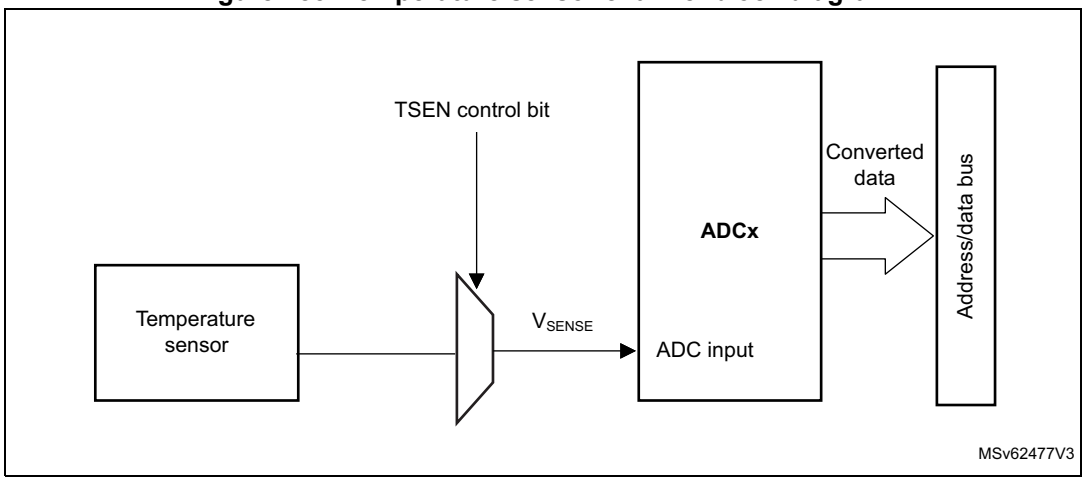
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel, which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 185 shows the block diagram of the temperature sensor.

Figure 185. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channel that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to the electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.

6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Caution: The above formula is valid only if the V_{REF+} condition specified in the table *Temperature sensor calibration values* of the datasheet are respected.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN, and TSEN bits should be set simultaneously.

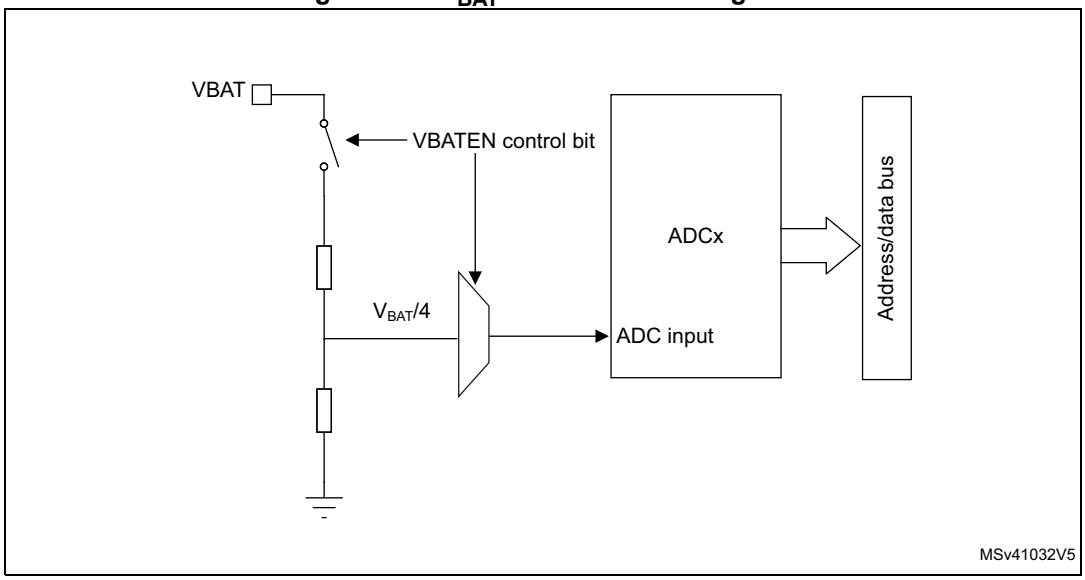
22.4.33 V_{BAT} supply monitoring

The VBATEN bit in the ADCC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage can be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 22.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 186 shows the block diagram of the V_{BAT} sensing feature.

Figure 186. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for $V_{BAT}/4$.

22.4.34 Monitoring the internal voltage reference

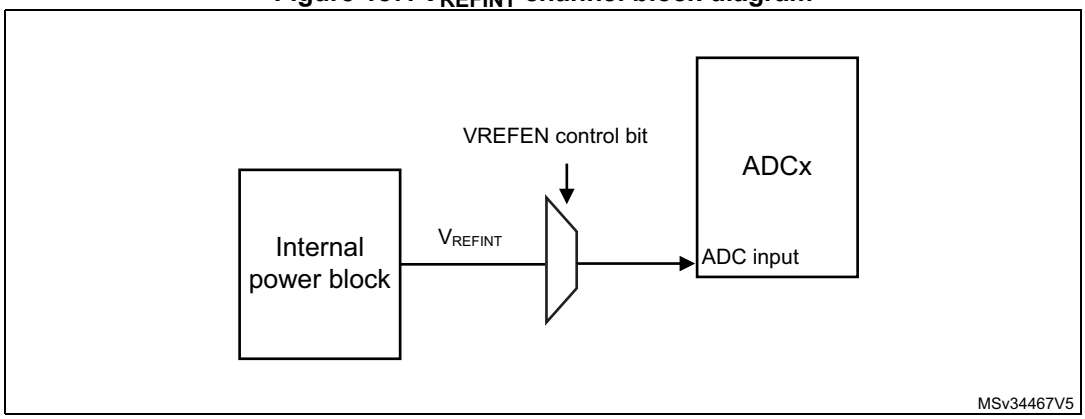
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 22.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 187 shows the block diagram of the V_{REFINT} sensing feature.

Figure 187. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value.
- ADC_DATA is the value measured by the ADC on channel x (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} - 1 = 4095$, or $2^8 - 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

22.4.35 Monitoring the DAC output voltage

The ADC is connected to a DAC internal channel. To measure the DAC output voltage, enable the DAC internal channel through DAC registers.

22.4.36 Monitoring the supply voltage

The ADCs are connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

22.5 ADC in low-power modes

Table 190. Effect of low-power modes on the ADC

Mode	Description
Sleep	No effect. DMA requests are functional. ADC interrupts cause the device to exit from Sleep mode.
Stop 0/1/2/3	The ADC is no more functional. The ADC must be powered down before entering Stop mode. The content of the ADC registers is kept.
Standby/Shutdown	The ADC is powered down and must be reinitialized after exiting Standby or Shutdown mode.

22.6 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2 and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)
- When the ADC internal voltage regulator output is ready (LDORDY flag)
- When the injected sequence context queue overflows (JQOVF flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 191. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	ADC internal voltage regulator ready	LDORDY	LDORDYIE			
	Injected context queue overflows	JQOVF	JQOVFIE			

22.7 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

22.7.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC internal voltage regulator output ready flag

This bit is set by hardware and cleared by software. It indicates that the ADC internal voltage regulator output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO internal voltage regulator disabled

1: ADC LDO internal voltage regulator enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an overflow of the injected queue of context occurs. It is cleared by software writing 1 to it. Refer to *Section 22.4.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Note: When the OVR flag is raised, the EOSMP flag is not generated correctly.

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

22.7.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: ADC internal voltage regulator interrupt enable

This bit is set and cleared by software to enable/disable the ADC internal voltage regulator interrupt.

0: Internal voltage regulator interrupt disabled

1: Internal voltage regulator interrupt enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the injected context queue overflow interrupt.

0: Injected context queue overflow interrupt disabled

1: Injected context queue overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

22.7.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs		rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to enter ADC calibration mode.

0: Normal mode

1: Calibration mode

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bit 28 **ADVREGEN**: ADC internal voltage regulator enable

This bits is set by software to enable the ADC internal voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC internal voltage regulator disabled

1: ADC internal voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 22.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Note: In dual ADC regular simultaneous mode and interleaved mode, the ADSTP bit of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

This bit is cleared by hardware:

– In single conversion mode when software trigger is selected (JEXTSEL=0x0 and JDISEN = 1): at the assertion of the end of injected conversion sequence (JEOS) flag.

– In all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1,ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

This bit is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1): at the assertion of the end of regular conversion sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0), except for ADVREGEN bit which must be set. In addition, the software must wait for the startup time of the voltage regulator.

22.7.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **JQDIS**: Injected queue disable

This bit is set and cleared by software to disable the injected queue mechanism:

0: Injected queue enabled

1: Injected queue disabled

Note: The software is allowed to write this bit only when ADEN = 0, ADSTART = 0, and JADSTART = 0.

Note: A set or reset of JQDIS bit causes the injected queue to be flushed and the ADC_JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10010: ADC analog input channel 18 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0, ADSTART = 0, and JADSTART = 0.

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the JAUTO bit of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: ADC_JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: ADC_JSQR mode 0: The Queue is never empty and maintains the last written configuration to ADC_JSQR.

1: ADC_JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 22.4.21: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (DUAL bits of ADCC_CCR register are not equal to zero), the JDISCEN bit of the slave ADC is no more writable and its content is equal to the JDISCEN bit of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCNUM[2:0] bits of the slave ADC are no more writable and their content is equal to the DISCNUM[2:0] bits of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bits of the slave ADC is no more writable and its content is equal to the DISCEN bit of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bits of the slave ADC is no more writable and its content is equal to the DISCEN bit of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bits of the slave ADC is no more writable and its content is equal to the DISCEN bit of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0

00001: adc_ext_trg1

00010: adc_ext_trg2

00011: adc_ext_trg3

00100: adc_ext_trg4

00101: adc_ext_trg5

00110: adc_ext_trg6

00111: adc_ext_trg7

...

11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only

01: DMA one-shot mode selected

10: ADF mode selected

11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

22.7.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw				rw	rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled

1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the SMPTRIG bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling and clear the BULB bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0, JADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: 1-bit shift
0010: 2-bit shift
0011: 3-bit shift
0100: 4-bit shift
0101: 5-bit shift
0110: 6-bit shift
0111: 7-bit shift
1000: 8-bit shift
1001: 9-bit shift
1010: 10-bit shift
Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **JOVSE**: Injected oversampling enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected oversampling disabled
1: Injected oversampling enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular oversampling enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled
1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

22.7.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

22.7.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **SMPx[2:0]:** Channel x sampling time selection (x = 18 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

22.7.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[18:16]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **PCSEL[18:0]**: Channel i ($V_{IN[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{IN[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{IN[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

22.7.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

22.7.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

22.7.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

22.7.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

22.7.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 22.4.26: Data management*.

22.7.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

22.7.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET_CH[4:0]					SSAT	USAT	POS OFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:0 Reserved, must be kept at reset value.

22.7.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:16]					
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

22.7.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G COMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

22.7.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 22.4.26: Data management*.

22.7.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWDCH[18:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWDCH[18:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

22.7.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[18:16]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWDCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWDCH[18:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[18:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

22.7.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1. Refer to *Section 22.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*.

22.7.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
rw	rw	rw							rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter
These bits are set and cleared by software.
000: No filtering, one detection generates an AWD1 flag or an interrupt
001: two consecutive detections generate an AWD1 flag or an interrupt
...
111: Eight consecutive detections generate an AWD1 flag or an interrupt
Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold
These bits are set and cleared by software to define the higher threshold for analog watchdog 1. Refer to *Analog windows watchdog* section.

22.7.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold
These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

22.7.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold
These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

22.7.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold
These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

22.7.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:16]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold
These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

22.7.27 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT[6:0]**: Calibration factors

These bits can be written by hardware or by software.

Once the calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is applied once a new conversion is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled, no conversion is ongoing).

22.7.28 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP0
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OP0:** Option bit 0

This bit is set and cleared by software to enable the V_{CORE} channel.

0: V_{CORE} disabled

1: V_{CORE} enabled

22.8 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

22.8.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
			r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
			r		r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LDORDY_SLV:** ADC internal voltage regulator flag of the slave ADC

This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected context queue overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY_MST:** ADC internal voltage regulator flag of the master ADC
This bit is a copy of the LDORDY bit in the corresponding ADC_ISR register.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected context queue overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

22.8.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.	Res.	Res.	Res.	Res.	Res.
							rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable
This bit is set and cleared by software to enable/disable the V_{BAT} channel.
0: V_{BAT} channel disabled
1: V_{BAT} channel enabled
Note: V_{BAT} is not available on all ADC instances. Refer to Section 22.4.4: ADC connectivity for details.

Bit 23 **TSEN**: Temperature sensor voltage enable
This bit is set and cleared by software to control V_{SENSE} channel.
0: Temperature sensor channel disabled
1: Temperature sensor channel enabled
Note: The temperature sensor is not available on all ADC instances. Refer to Section 22.4.4: ADC connectivity for details.
Note: The software is allowed to write this bit only when ADEN = 0.

Bit 22 **VREFEN**: V_{REFINT} enable
This bit is set and cleared by software to enable/disable the V_{REFINT} channel.
0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled
Note: V_{REFINT} is not available on all ADC instances. Refer to Section 22.4.4: ADC connectivity for details.
Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format

This bitfield are set and cleared by software. It specifies the data format in the common data register ADCC_CDR and ADCC_CDR2.
00: Dual ADC mode without data packing (ADCC_CDR and ADCC_CDR2 registers not used).
01: Reserved
10: Data formatting mode for any data width (ADCC_CDR data register is used when the data width is less than 16 bits, otherwise ADCC_CDR2 register is used)
11: Data formatting mode for data width lower that 8 bits (ADCC_CDR data register is used)

Note: The software is allowed to write these bits only when ADEN = 0 (which ensures that no regular conversion is ongoing).

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between two sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 189* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode
The following settings apply to dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + Injected simultaneous mode
00010: Combined regular simultaneous + Alternate trigger mode
00011: Combined interleaved mode + Injected simultaneous mode
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

22.8.3 ADC common regular data register for dual mode (ADCC_CDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 22.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT, POSOFF)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 22.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT, POSOFF)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

22.8.4 ADC common regular data register for dual mode (ADCC_CDR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_ALT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA_ALT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs.

In dual mode, these bits contain the regular 32-bit data of the master and the slave ADC. Refer to *Section 22.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT, POSOFF)*)

22.9 ADC register map

ADC1 and ADC2 are master and slave ADC, respectively.

Table 192. ADC register map and reset values for each ADC

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDYIE	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	Res.	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0		1	0																							0	0	0	0	0	0
0x00C	ADC_CFGR1	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]		DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]		
	Reset value	0x8000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0				0	0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]				SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	Res.	Res.	Res.	SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved	Reserved																															
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 192. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044-0x048	Reserved	Reserved																															
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	ADC_OFCFGR1	OFFSET1_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2	OFFSET2_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3	OFFSET3_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4	OFFSET4_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074-0x07C	Reserved	Reserved																															
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 192. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090-0x09C	Reserved	Reserved																															
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[18:0]																		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		LTR[22:0]																					
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		HTR[22:0]																					
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	Reserved	Reserved																															
0x0C4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved	Reserved																															
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP0
	Reset value																																0
0x0D4-0x0FC	Reserved	Reserved																															

Table 193. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	LDORDY_SLV	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	LDORDY_MST	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					slave ADC2																			master ADC1									
	Reset value				0		0	0	0	0	0	0	0	0	0	0	0				0		0	0	0	0	0	0	0	0	0	0	0
0x004	Reserved	Res.																															
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.	Res.	Res.	Res.	Res.	Res.	DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0							0	0			0	0	0	0				0	0	0	0	0

Table 193. ADC register map and reset values (master and slave ADC common registers) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	ADCC_CDR	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	ADCC_CDR2	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved	Reserved																															

Refer to *Section 2.3* for the register boundary addresses.

23 Digital-to-analog converter (DAC)

23.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DACx_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

23.2 DAC main features

The DAC main features are the following (see *Figure 188: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Autonomous mode to reduce the power consumption for the system
- Input voltage reference from VREF+ pin or internal VREFBUF reference

Figure 188 shows the block diagram of a DAC channel and *Table 195* gives the pin description.

23.3 DAC implementation

Table 194. DAC features

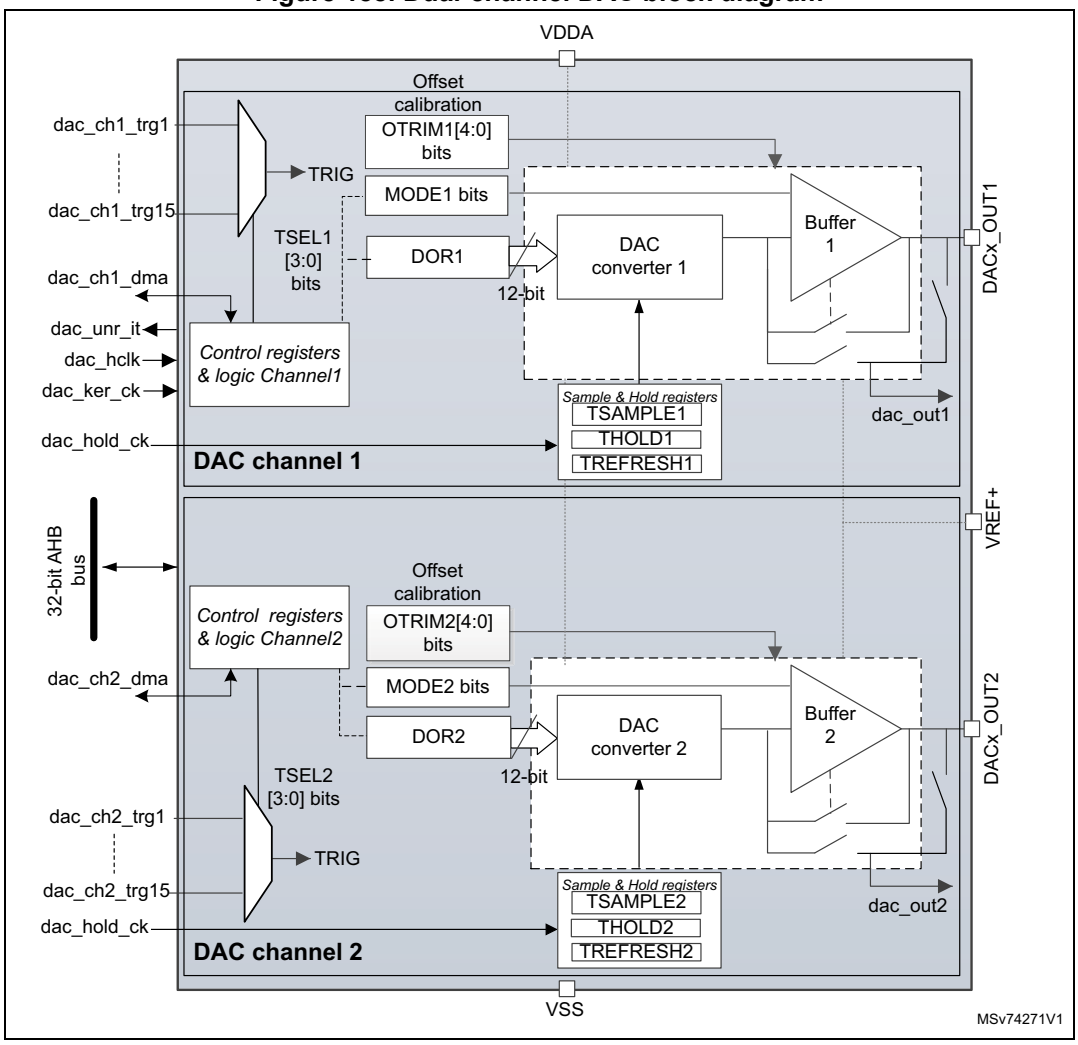
DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 Msps
Autonomous mode(1)	X

1. The autonomous mode is supported in Stop 1 mode.

23.4 DAC functional description

23.4.1 DAC block diagram

Figure 188. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to *Section 23.3: DAC implementation* for channel2 availability.

23.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 195. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 196. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 197. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim4_trgo	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	Reserved	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg7 (x = 1, 2)	Reserved	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	tim15_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_ch1	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	lptim3_ch1	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

23.4.3 DAC clocks

Two clock sources can be used to update the DAC:

- dac_hclk: DAC peripheral clock (AHB clock)
- dac_ker_ck: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- dac_hold_ck: low-power clock used in sample and hold mode

The DAC clock is selected in the RCC.

23.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

23.4.5 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

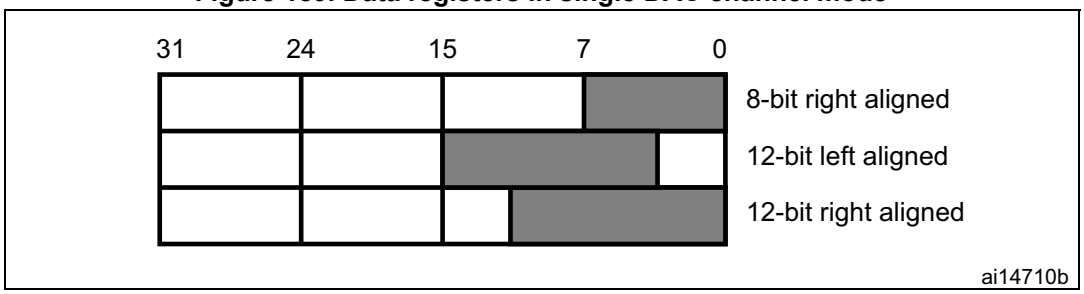
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 189. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

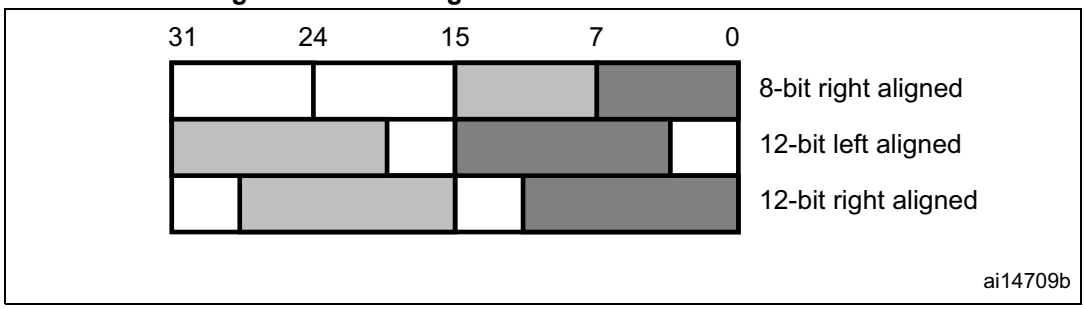
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 190. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2's complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 198. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

23.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk or dac_ker_ck clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

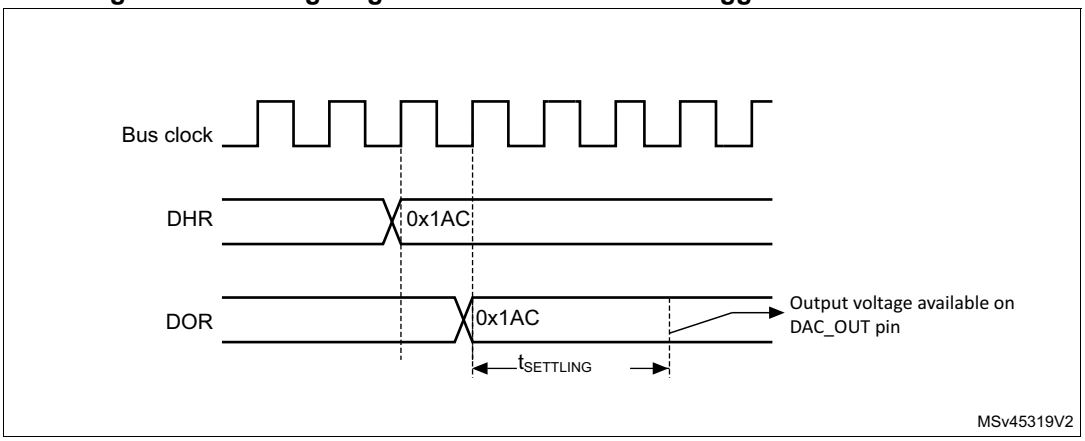
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 199. HFSEL description

HFSEL [1:0]	Clock frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
00	< 48 MHz	3	4	DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.
01	≥ 48 MHz(1)	5	5	DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.
11	Reserved	-	-	-

1. Refer to the device datasheet for the value of the maximum dac_hclk or dac_ker_ck frequency.

Figure 191. Timing diagram for conversion with trigger disabled TEN = 0



23.4.7 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DAC output} = V_{REF} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

23.4.8 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

23.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

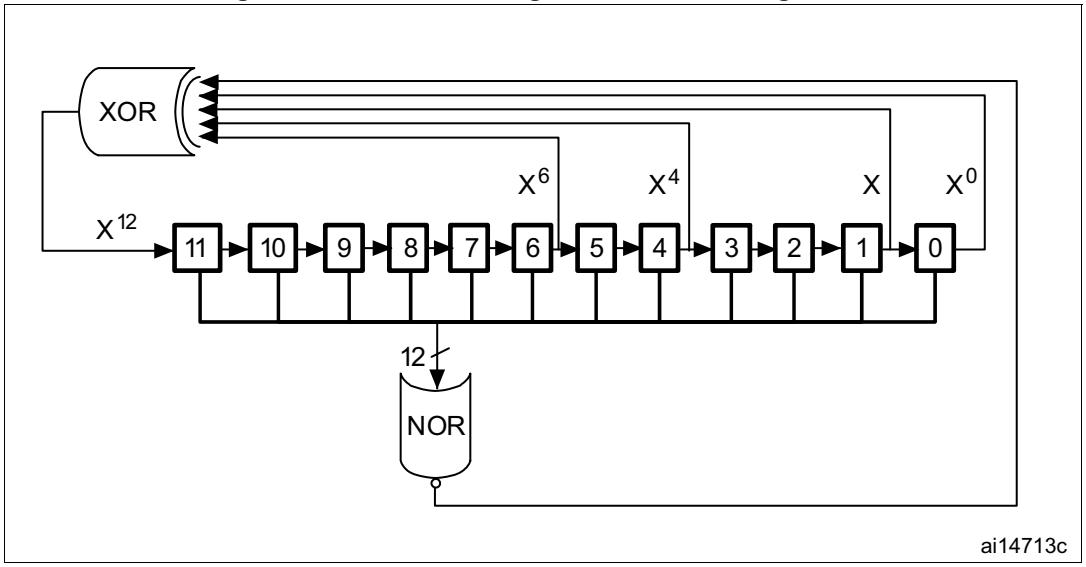
The following conditions must be met to change from double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

23.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 192. DAC LFSR register calculation algorithm

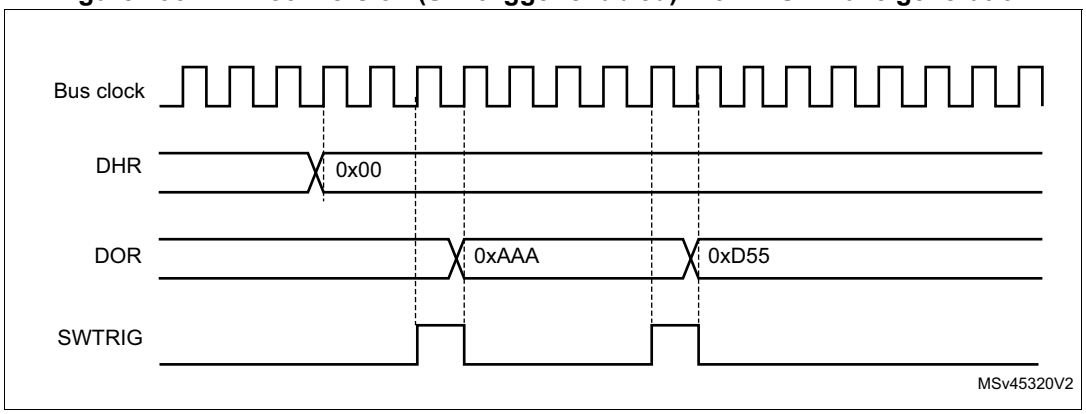


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 193. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

23.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 194. DAC triangle wave generation

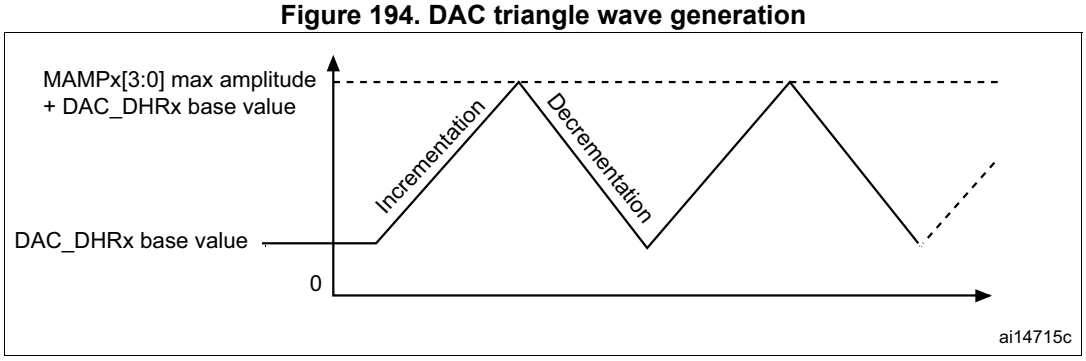
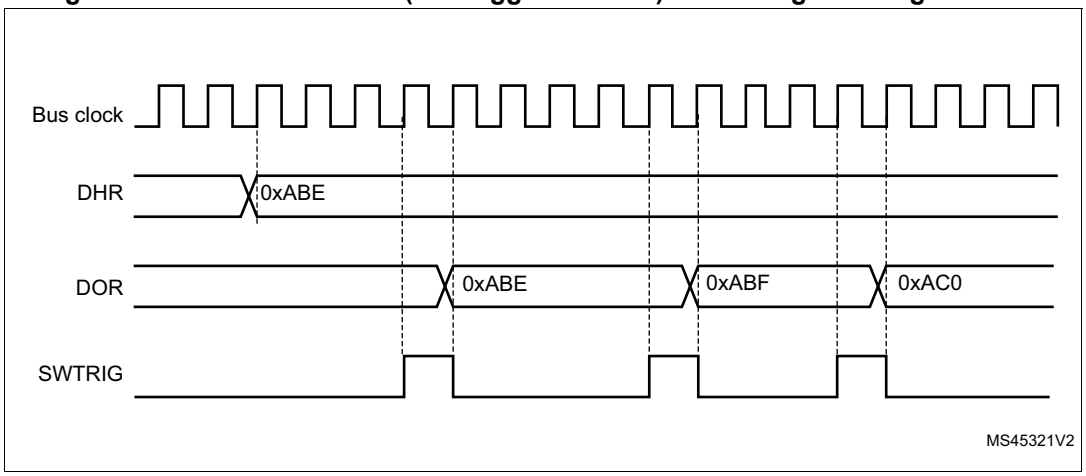


Figure 195. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

23.4.12 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 200. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{REF+} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms
(where R_{BON} = 2 kΩ)

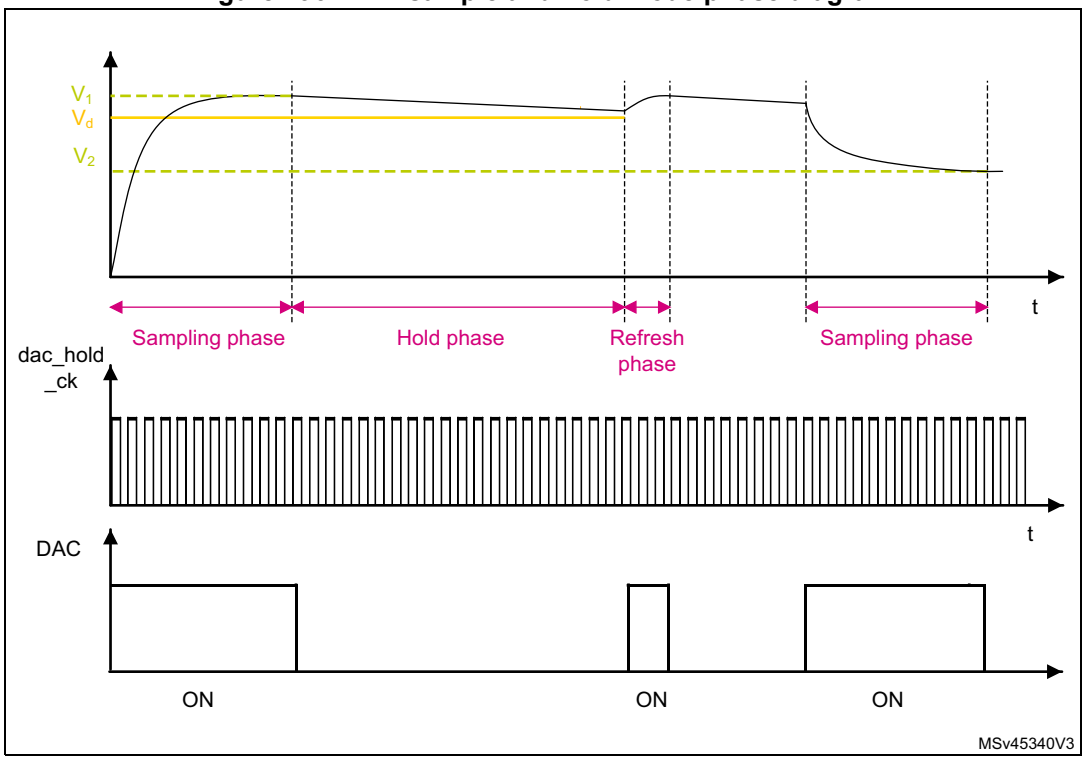
Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs
(where N_{LSB} = 10 (10 LSB drop during the hold phase)

Hold phase:

$D_v = i_{leak} * t_{hold} / C_{SH}$ = 0.0073 V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 196. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Table 201. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

23.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left((D / 2^N) \times G \times V_{REF}\right) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REF} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{REF+}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REF+} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby mode) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) Until the CAL_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REF+} and temperature conditions do not change during device operation while it enters more often in Standby mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

23.4.14 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC_DORx (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bits.
2. Configure the DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHRx register and transfers it into DAC_DOR1 (three AHB clock cycles later).

23.4.15 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

23.4.16 DAC autonomous mode

The autonomous mode can be used to update the DAC output voltage in Stop mode. This allows DMA transfers to be performed when the device operates in Run, Sleep or Stop mode. The autonomous mode is supported AUTOMODE only when the DAC is in sample and hold mode. It is enabled by setting the bit in the DAC_AUTOCR register. DMA requests must also be enabled (DMAEN = 1).

When the AUTOMODE bit is set, each hardware trigger signal generates an AHB clock request to the RCC. Once the peripheral receives the AHB clock, the content of the DAC_DHRx register is loaded into DAC_DOR register and a DMA request is generated. When the DMA transaction is complete, the DAC deasserts the AHB clock request and waits for a new trigger event.

When the sample and hold mode is selected, the dac_ker_ck low-power clock must be enabled by the RCC.

DMA transfers in autonomous mode are performed following the sequence described below:

1. The DAC outputs the voltage on a GPIO or on an on-chip peripheral (dac_ker_ck and dac_hclk are not required).
2. The DAC receives a hardware trigger from another peripheral.
3. The DAC asserts the clock request.
4. The DAC loads the DAC_DHRx register into DAC_DOR register.
5. The DAC generates a DMA request.
6. The DMA transfers the new data from memory to DAC_DHRx registers.
7. The DAC deasserts the clock request.

To initialize the DAC to operate autonomously in Stop mode, follow the sequence below:

1. Enable the DAC clock autonomous mode in the RCC.
2. Select the DAC kernel clock in the RCC.
3. Enable dac_hold_ck clock for the DAC in the RCC.
4. Select a sample and hold mode through MODEx bit.
5. Configure the sample and hold mode by setting TSAMPLEx[9:0], THOLDx[9:0], TREFRESHx[7:0] bits.
6. Load the target value into DAC_DHRx register.
7. Set TENx bit to output the DAC value.
8. Configure the trigger setting through TSEL[3:0] bits.
9. Set DMAENx bit and configure the DMA interface.
10. Set the AUTOMODE bit.

Timing uncertainty of DAC_DHRx to DAC_DOR transfer

After each trigger event, the data contained in DAC_DHRx register is transferred to the DAC_DOR register. Since this transfer is based on AHB clock, the transfer timing depends on the clock availability.

When the AHB clock is stopped, a minimum time is required to restart it. This time depends on the clock source.

23.5 DAC in low-power modes

Table 202. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA
Stop 0[1]	The DAC remains functional and can perform DMA transfers in sample and hold mode.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

1. Refer to *Section 23.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

23.6 DAC interrupts

Table 203. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

23.7 DAC registers

Refer to *Section 1 on page 87* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

23.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable
This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled
Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095
Note: These bits are available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable
These bits are set/reset by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection
These bits select the external event used to trigger DAC channel2
0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15
Refer to the trigger selection tables in for details on trigger configuration and mapping.
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register

1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.

These bits are available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable
These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in for details on trigger configuration and mapping.
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register
Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

23.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

23.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

23.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

23.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

23.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

23.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

23.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

23.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

23.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

23.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

23.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

23.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

23.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_ FLAG2	DMAU DR2	DORST AT2	DAC2R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_ FLAG1	DMAU DR1	DORST AT1	DAC1R DY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

23.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

23.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[1:0]		Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA normal mode selected

1: DMA double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 23.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 23.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
00: High frequency interface mode disabled
01: High frequency interface mode enabled for AHB clock frequency > 80 MHz
10: High frequency interface mode enabled for AHB clock frequency >160 MHz
11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

23.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

23.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

23.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

23.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 23.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

23.7.21 DAC autonomous mode control register (DAC_AUTOCR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AUTO MODE	Res.	Res.	Res.	Res.	Res.	Res.
									rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **AUTOMODE:** DAC autonomous mode
This bit is set and cleared by software.
0: DAC autonomous mode disabled
1: DAC autonomous mode enabled

Bits 21:0 Reserved, must be kept at reset value.

23.7.22 DAC register map

Table 204 summarizes the DAC registers.

Table 204. DAC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DAC_CR**	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:1]			TSEL2[0]	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:1]			TSEL1[0]	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**DAC_ SWTRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	**DAC_ DHR12R1**	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**DAC_ DHR12L1**	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	**DAC_ DHR8R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 204. DAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	**DAC_ DHR12R2**	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**DAC_ DHR12L2**	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	**DAC_ DHR8R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**DAC_ DHR12RD**	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**DAC_ DHR12LD**	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	**DAC_ DHR8RD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**DAC_ DOR1**	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	**DAC_ DOR2**	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x34	**DAC_SR**	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL[1]	HFSEL[0]	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0					0	0						0	0	0

Table 204. DAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	**DAC_ SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_ SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50	Reserved	Res.																															
0x54	**DAC_ AUTOCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AUTOMODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0																						
0x58-0x60	Reserved	Res.																															
0x64-0x68	Reserved	Res.																															

Refer to *Section 2.3* for the register boundary addresses.

24 Voltage reference buffer (VREFBUF)

24.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs and DACs, and also as voltage reference for external components through the VREF+ pin.

24.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

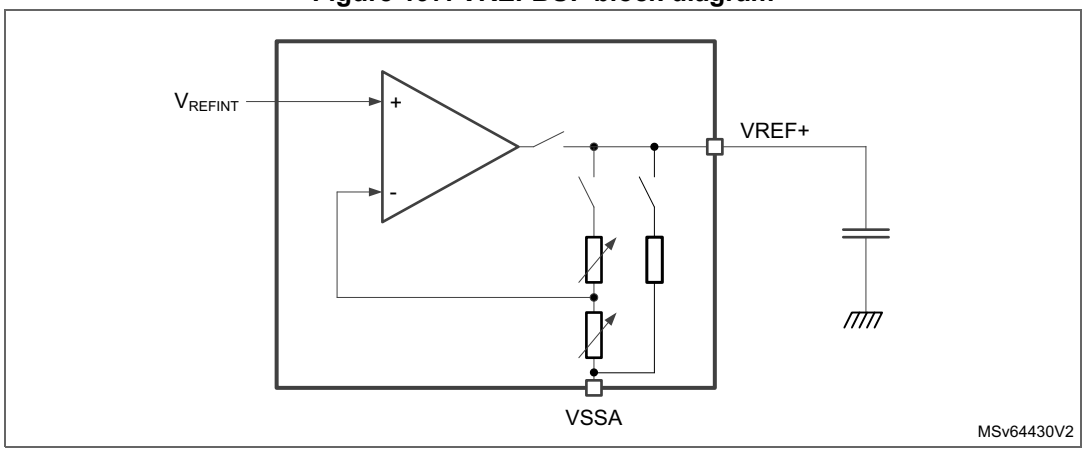
Table 205. VREFBUF typical values

Symbol	Value
VREFBUF0	1.5 V
VREFBUF1	1.8 V
VREFBUF2	2.048 V
VREFBUF3	2.5 V

Note: Refer to the product datasheet for more details.

24.3 VREFBUF functional description

Figure 197. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports four voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected
- VRS = 001: VREFBUF1 voltage selected
- VRS = 010: VREFBUF2 voltage selected
- VRS = 011: VREFBUF3 voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 206. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

24.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

24.5 VREFBUF registers

24.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.
VRS = 001: VREFBUF1 voltage selected.
VRS = 010: VREFBUF2 voltage selected.
VRS = 011: VREFBUF3 voltage selected.
Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.
0: V_{REF+} pin is internally connected to the voltage reference buffer output.
1: V_{REF+} pin is high impedance.
Refer to *Table 206: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

24.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the flash memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the flash memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

24.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 207. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

25 Comparator (COMP)

25.1 Introduction

The device embeds two ultra-low-power comparators COMP1 and COMP2.

These comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

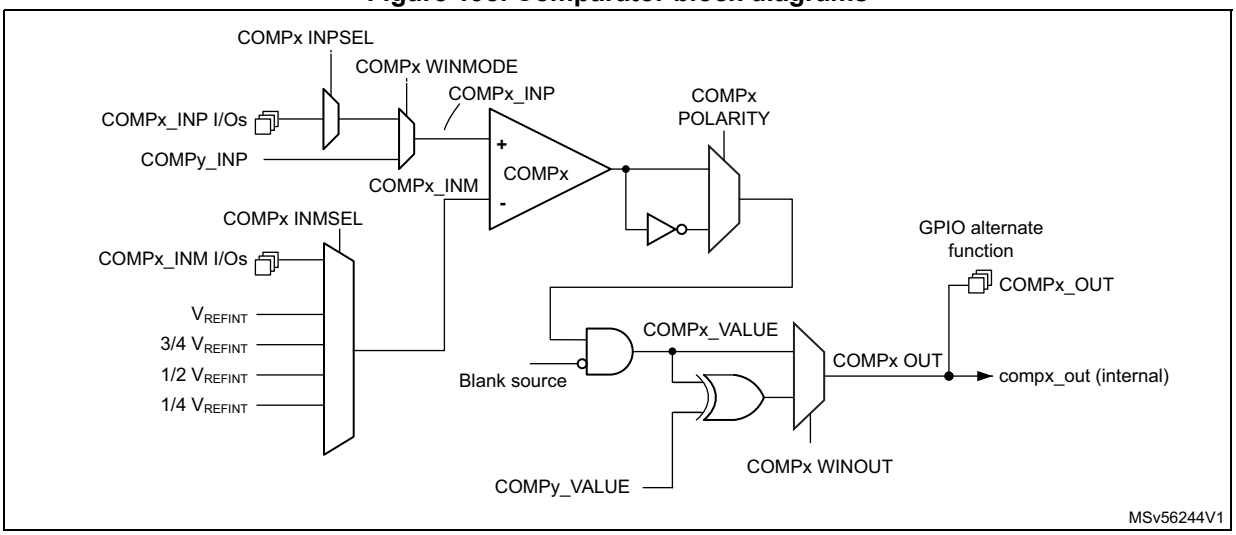
25.2 COMP main features

- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by a scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed/consumption
- Outputs that can be redirected to an I/O or to timer inputs for triggering break events for fast PWM shutdowns
- Comparator outputs with blanking source
- Comparators that can be combined as a window comparator
- Interrupt generation capability for each comparator with wake-up from Sleep and Stop modes (through the EXTI controller)

25.3 COMP functional description

25.3.1 COMP block diagram

Figure 198. Comparator block diagrams



25.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF_CLR inputs
- Input capture for timing measures

The comparator output can be simultaneously redirected internally and externally.

Table 208. COMP1 noninverting input assignment

COMP1_INP	COMP1_INPSEL[1:0]
COMP1_INP1	00
COMP1_INP2	01
COMP1_INP3	10
COMP1_INP4	11

Table 209. COMP1 inverting input assignment

COMP1_INM	COMP1_INMSEL[3:0]
¼ V_{REFINT}	0000
½ V_{REFINT}	0001
¾ V_{REFINT}	0010
V_{REFINT}	0011
DAC channel1	0100
DAC channel2	0101
COMP1_INM1	0110
COMP1_INM2	0111
Reserved	≥ 1000

Table 210. COMP2 noninverting input assignment

COMP2_INP	COMP2_INPSEL[1:0]
COMP2_INP1	00
COMP2_INP2	01
Reserved	10
Reserved	11

Table 211. COMP2 inverting input assignment

COMP2_INM	COMP2_INMSEL[3:0]
¼ V_{REFINT}	0000
½ V_{REFINT}	0001
¾ V_{REFINT}	0010
V_{REFINT}	0011
DAC channel1	0100
DAC channel2	0101
COMP2_INM1	0110
COMP2_INM2	0111
Reserved	≥ 1000

Table 212. COMP1 output-blanking PWM assignment

PWM output	COMP1_BLANKSEL[4:0]
None (no blanking)	00000
tim1_oc5	xxxx1
tim2_oc3	xxx1x

Table 212. COMP1 output-blanking PWM assignment (continued)

PWM output	COMP1_BLANKSEL[4:0]
tim3_oc3	xx1xx
Reserved	Others

Table 213. COMP2 output-blanking PWM assignment

PWM output	COMP2_BLANKSEL[4:0]
None (no blanking)	00000
tim3_oc4	xxxx1
tim15_oc1	xx1xx
Reserved	Others

25.3.3 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, the comparator programming must not be altered in case of spurious register access or program counter corruption. For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, the COMPxLOCK bit can be set to 1. This causes the whole COMPx_CSR register to become read-only, including the COMPxLOCK bit.

The write protection can be reset only by an MCU reset.

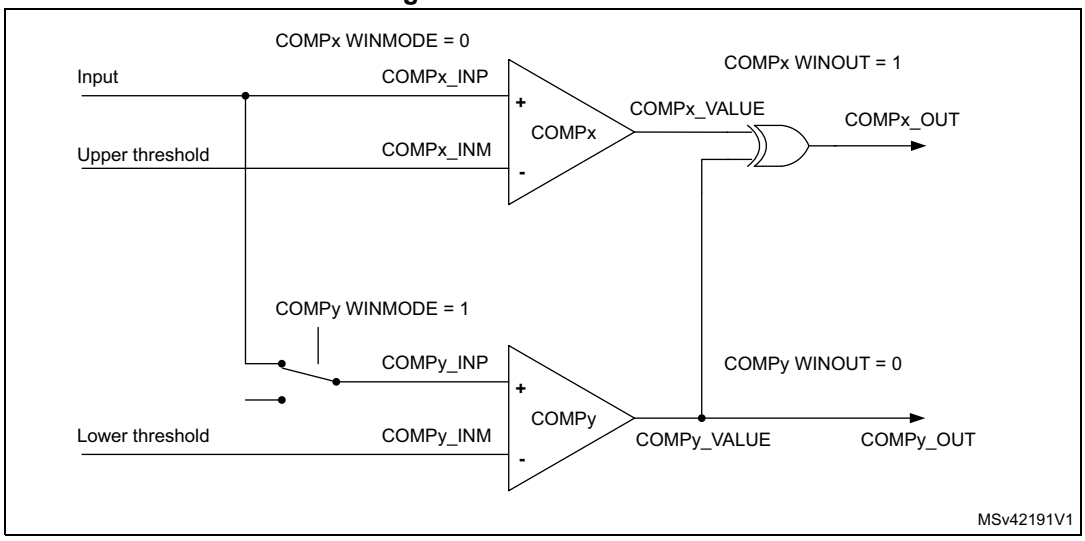
25.3.4 Window comparator

The purpose of the window comparator is to monitor the analog voltage if it is within the voltage range defined by the lower and upper thresholds.

The two embedded comparators can be used to create a window comparator. The monitored analog voltage is connected to the noninverting (plus) inputs of the two comparators connected together. The upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators.

Two noninverting inputs can be connected internally by enabling the WINMODE bit to save one I/O for other purposes.

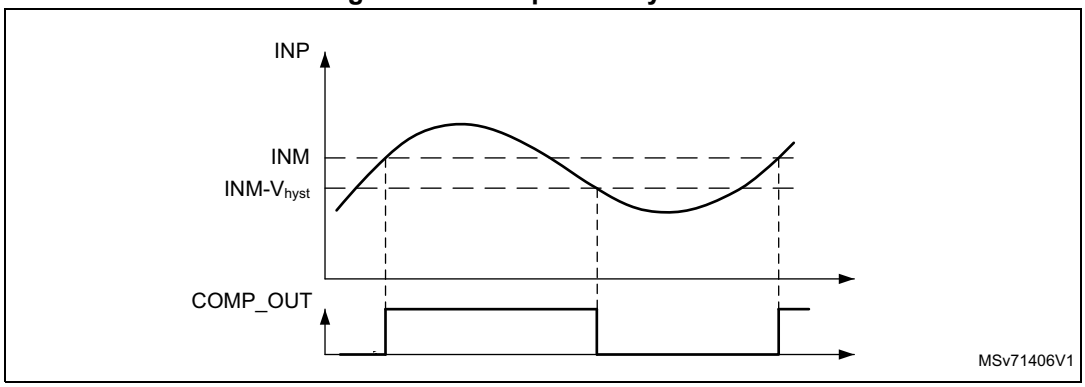
Figure 199. Window mode



25.3.5 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting a low-power mode) to be able to force the hysteresis value using external components.

Figure 200. Comparator hysteresis

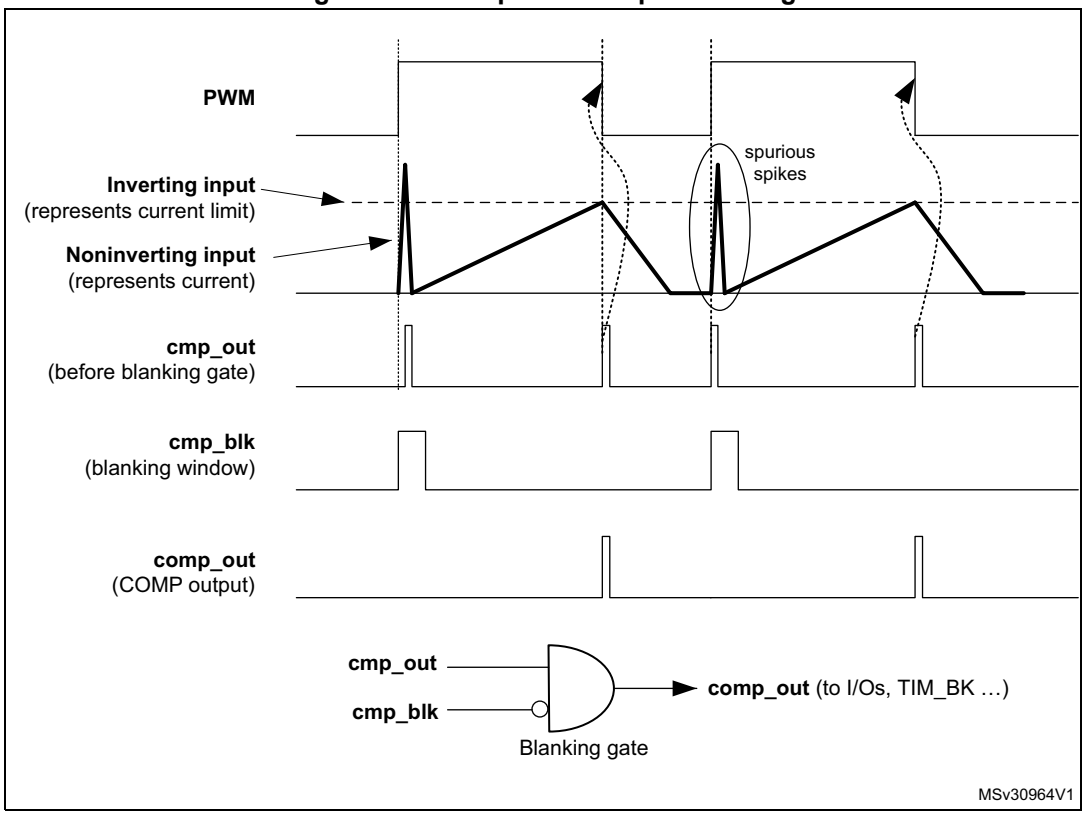


25.3.6 Comparator output-blanking function

The blanking function prevents the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). This blanking function consists of a selection of a blanking window that is a timer output compare signal. The selection is done by the software (refer to the comparator register description for possible blanking signals).

The complementary of the blanking signal is AND-ed with the comparator output to provide the wanted comparator output (see the example in *Figure 201*).

Figure 201. Comparator output blanking



25.3.7 COMP power and speed modes

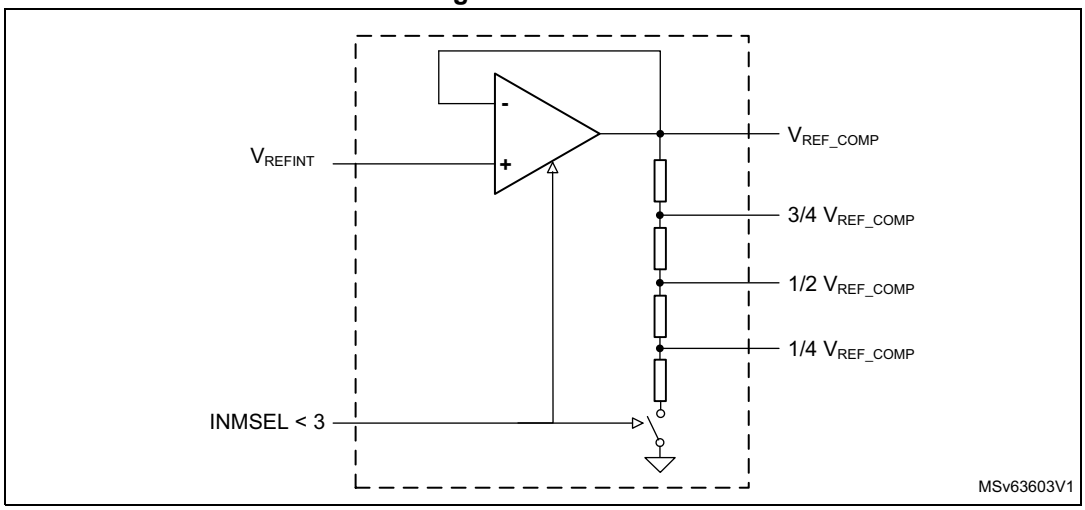
COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

25.3.8 Scaler function

The scaler block provides the different voltage reference levels to the comparator inputs. This block is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference.The amplifier and the resistor bridge are enabled by setting the INMSEL value in the COMP_CFGRx registers, to connect the corresponding inverting input to the scaler output.

When the divided voltage is not used, the resistor bridge and the amplifier are disabled to reduce the consumption. When the resistor bridge is disconnected, the 1/4 VREF_COMP, 1/2 VREF_COMP, and 3/4 VREF_COMP levels are equal to VREF_COMP.

Figure 202. Scaler



25.4 COMP in low-power modes

Table 214. Comparator behavior in the low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit Stop mode.
Standby	COMP registers are powered down and must be reinitialized after exiting Standby mode.

25.5 COMP interrupts

The comparator outputs are internally connected to the extended interrupts and events controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit the low-power modes.

Refer to *Section 17: Extended interrupts and event controller (EXTI)* for more details.

To enable the COMPx interrupt, follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity.
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines.
3. Enable the COMPx.

Table 215. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
COMP1 output	In EXTI	Through EXTI	Yes	Yes	No
COMP2 output	In EXTI	Through EXTI	Yes	Yes	No

25.6 COMP registers

25.6.1 COMP1 control and status register (COMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]	
rs	r						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** COMP1_CSR register lock

This bit is set by the software and cleared by reset. It locks the whole content of COMP1_CSR.

0: COMP1_CSR read/write bits can be written by software.

1: COMP1_CSR bits can be read but not written by software.

Bit 30 **VALUE:** COMP1 output status

This bit is read-only. It reflects the level of the COMP1 output after the polarity selector and blanking (see *Figure 201: Comparator output blanking*).

Bits 29:25 Reserved, must be kept at reset value.

Bits 24:20 **BLANKSEL[4:0]:** COMP1 blanking source selector

This bitfield is controlled by software (if not locked), and selects the blanking source.

00000: None (no blanking)

00001: tim1_oc5

00010: tim2_oc3

Others: Reserved

Bits 19:18 **PWRMODE[1:0]:** COMP1 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and, as a consequence, the speed of the COMP1.

00: High speed

01: Intermediate speed and power

10: Medium speed and power

11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP1 hysteresis selector
This bitfield is controlled by software (if not locked). It selects the COMP1 hysteresis.
00: None
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** COMP1 polarity selector
This bit is controlled by software (if not locked). It selects the COMP1 output polarity.
0: Noninverted
1: Inverted

Bit 14 **WINOUT:** COMP1 output selector
This bit is controlled by the software (if not locked). It selects the COMP1 output.
0: COMP1_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 199: Window mode*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP1 noninverting input selector for window mode
This bit is controlled by software (if not locked). It selects the signal for the COMP1_INP input of the COMP1.
0: Signal selected with INPSEL[1:0]
1: COMP2_INP signal of COMP2 (required for window mode, see *Figure 199: Window mode*)

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMP1 signal selector for noninverting input
This bitfield is controlled by software (if not locked). It selects the signal for the noninverting input COMP1_INP (see *Table 210: COMP2 noninverting input assignment* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP1 signal selector for inverting input INM
This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP1_INM (see *Table 209: COMP1 inverting input assignment* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP1 enable
This bit is controlled by the software (if not locked). It enables COMP1.
0: COMP1 disabled
1: COMP1 enabled

25.6.2 COMP2 control and status register (COMP2_CSR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]	
rs	r						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** COMP2_CSR register lock

This bit is set by software and cleared by reset. It locks the whole content of COMP2_CSR.
0: COMP2_CSR read/write bits can be written by software.
1: COMP2_CSR bits can be read but not written by software.

Bit 30 **VALUE:** COMP2 output status

This bit is read-only. It reflects the level of the COMP2 output after the polarity selector and blanking (see *Figure 201: Comparator output blanking*).

Bits 29:25 Reserved, must be kept at reset value.

Bits 24:20 **BLANKSEL[4:0]:** COMP2 blanking source selector

This bitfield is controlled by software (if not locked), and selects the blanking source.
00000: None (no blanking)
00001: tim3_oc4
Others: Reserved

Bits 19:18 **PWRMODE[1:0]:** COMP2 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and, as a consequence, the speed of the COMP2.
00: High speed
01: Intermediate speed and power
10: Medium speed and power
11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP2 hysteresis selector

This bitfield is controlled by software (if not locked). It selects the COMP2 hysteresis.
00: None
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** COMP2 polarity selector

This bit is controlled by software (if not locked). It selects the COMP2 output polarity.
0: Noninverted
1: Inverted

Bit 14 **WINOUT:** COMP2 output selector

This bit is controlled by software (if not locked). It selects the COMP2 output.
0: COMP2_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 199: Window mode*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP2 noninverting input selector for window mode

This bit is controlled by software (if not locked). It selects the signal for the COMP2_INP input of the COMP2.

0: Signal selected with INPSEL[1:0]

1: COMP1_INP signal of COMP1 (required for window mode, see *Figure 199: Window mode*)

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMP2 signal selector for noninverting input

This bitfield is controlled by software (if not locked). It selects the signal for the noninverting input COMP2_INP (see *Figure 210: COMP2 noninverting input assignment* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP2 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP2_INM (see *Figure 209: COMP1 inverting input assignment* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP2 enable

This bit is controlled by software (if not locked). It enables COMP2.

0: COMP2 disabled

1: COMP2 enabled

25.6.3 COMP register map

Table 216. COMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]		POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INPSEL[3:0].				Res.	Res.	Res.	EN
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0				0
0x04	COMP2_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]		POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INPSEL[3:0].				Res.	Res.	Res.	EN
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0				0

Refer to *Section 2.3* for the register boundary addresses.

26 Operational amplifier (OPAMP)

26.1 OPAMP introduction

STM32U3 series devices embed two operational amplifiers with two inputs and one output each. The three I/Os can be connected to the external pins, this enables any type of external interconnections. The operational amplifier can be configured internally as a follower or as an amplifier with a noninverting gain ranging from 2 to 16.

The positive input can be connected to the internal DAC.

The output can be connected to the internal ADC.

26.2 OPAMP main features

- Rail-to-rail input voltage range
- Low input bias current
- Low input offset voltage
- Low-power mode
- High-speed mode to achieve a better slew rate
- Fast wake-up time
- Gain bandwidth of 1.6 MHz

26.3 OPAMP functional description

The OPAMP has several modes.

Each OPAMP can be individually enabled, when disabled the output is high-impedance.

When enabled, it can be in calibration mode, all input and output of the OPAMP are then disconnected, or in functional mode.

There are three functional modes: the low-power mode, the high-speed mode, and the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in the *Section 26.3.3*.

26.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in RCC_APB1ENR1, and OPAMPSLPEN bit in RCC_APB1SLPENR1).

The OPAEN bit in OPAMPx_CSR enables and disables the OPAMP operation. The OPAMP registers configurations can be changed when the OPAEN bit is set in OPAMPx_CSR. However, it can create spurious effects (noise, glitch, overshoot, or saturation). If the configuration is changed, the application firmware must take care of these spurious effects (such as ignore the ADC result on the OPAMP output).

When the OPAMP output is no more needed, the OPAMP can be disabled to save power. All the configurations previously set (including the calibration) are maintained while the OPAMP is disabled.

26.3.2 Initial configuration

The OPAMP default configuration is a functional mode where the three IOs are connected to external pins. In the default mode, the OPAMP uses the factory trimming values (see “electrical characteristics” section of the datasheet for factory trimming conditions. Usually the temperature is 30 °C and the voltage is 3 V). The trimming values can be adjusted (see *Section 26.3.5*). The default configuration uses the normal mode that provides the standard performance. The OPALPM bit in OPAMPx_CSR can be set in order to switch the OPAMP to low-power mode and reduced performance. Normal, low-power, and high-speed modes characteristics are defined in the “electrical characteristics” section of the datasheet. Before utilization, the OPA_RANGE bit in OPAMP1_CSR must be set to 1.

As soon as the OPAEN bit in OPAMPx_CSR is set, the OPAMP is functional. The two input pins and the output pin are connected as defined in *Section 26.3.3* and the default connection settings can be changed.

Note: The inputs and output pins must be configured in analog mode (default state) in the corresponding GPIOx_MODER register.

26.3.3 Signal routing

The routing for the OPAMP pins is determined by the OPAMPx_CSR register.

The connections of the operational amplifiers (OPAMP1 and OPAMP2) are described in the table below.

Table 217. Operational amplifier possible connections

Signal	Pin	Internal	comment
OPAMP1_VINM	PA1	ADC1_IN4, OPAMP1_OUT or PGA	Controlled by OPAMODE and VM_SEL
OPAMP1_VINP	PA0	ADC1_IN3, dac1_out1	Controlled by bit VP_SEL
OPAMP1_VOUT	PA3	ADC1_IN6 COMP1_INP4	The pin is connected when the OPAMP is enabled. The ADC input is controlled by the ADC.
OPAMP2_VINM	PA7	ADC1_IN10, ADC2_IN6, OPAMP2_OUT or PGA	Controlled by bits OPAMODE and VM_SEL
OPAMP2_VINP	PA6	ADC1_IN9, ADC2_IN5, dac1_out2	Controlled by bit VP_SEL
OPAMP2_VOUT	PB0	ADC1_IN13 ADC2_IN9	The pin is connected when the OPAMP is enabled. The ADC input is controlled by the ADC.

26.3.4 OPAMP modes

The OPAMP inputs and outputs are all accessible on terminals. The amplifiers can be used in the following configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

The amplifier output pin is directly connected to the output pad to minimize the output impedance. It cannot be used as a general purpose I/O, even if the amplifier is configured as a PGA and only connected to the ADC channel.

The impedance of the signal must be maintained below a level that avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Refer to the “electrical characteristics” section in the datasheet for further details.

Standalone mode (external gain setting mode)

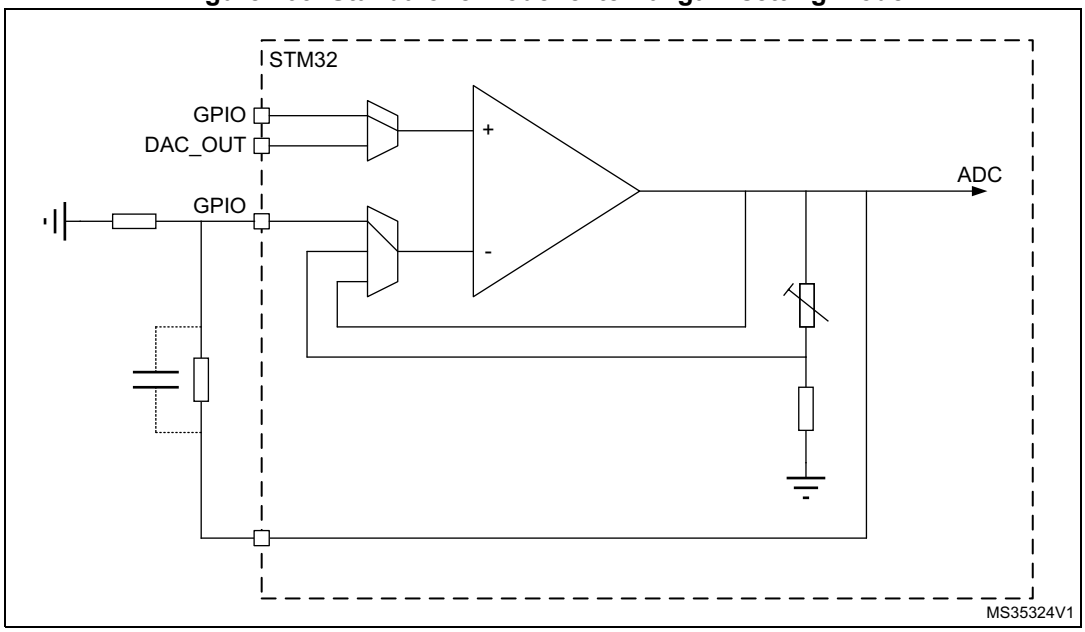
The procedure to use the OPAMP in standalone mode is detailed below:

1. Keep default value of OPAMPx_CSR and the default state of GPIOx_MODER.
2. As soon as OPAEN is set in OPAMPx_CSR, the two input pins and the output pin are connected to the operational amplifier.

This default configuration uses the factory trimming values and operates in normal mode (standard performance). The OPAMP behavior can be changed with the following bits in OPAMPx_CSR:

- If OPALPM is set to 1, the OPAMP switches in low-power mode in order to save power.
- If OPAHSM is set to 1, the OPAMP switches in high-speed mode in order to have a high slew rate.
- If USERTRIM is set to 1, the input offset values can be trimmed.

Figure 203. Standalone mode: external gain setting mode



Follower configuration mode

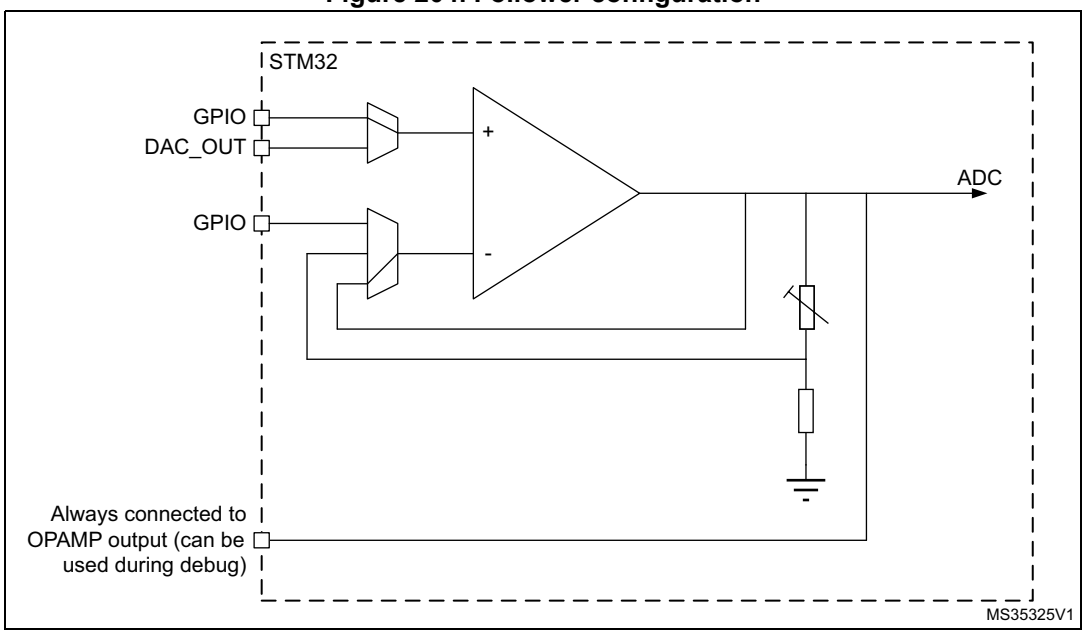
The procedure to use the OPAMP in follower mode is detailed below (all bits in OPAMPx_CSR):

1. Set OPAMODE[1:0] = 11 (internal follower).
2. Clear VP_SEL to 0 (GPIO connected to OPAMPx_VINP, named VINP in this document).
3. As soon as OPAEN is set to 1, the signal on the VINP pin is copied to the OPAMP_VOUT pin.

Note: The pin corresponding to OPAMP_VINM is free for another usage.

The signal on the OPAMP output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming that the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 204. Follower configuration



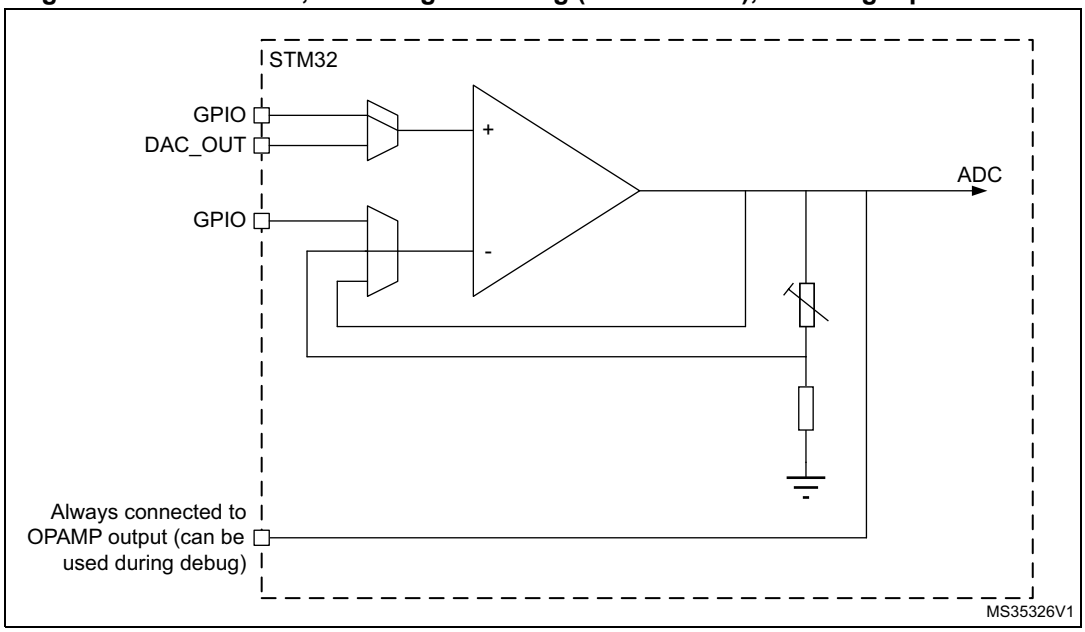
Programmable gain amplifier mode

The procedure to use the OPAMP to amplify the amplitude of an input signal is detailed below (all bits in OPAMPx_CSR):

1. Set OPAMODE[1:0] to 10 (internal PGA enabled).
2. Set PGA_GAIN[1:0] to the selected PGA gain (2, 4, 8 or 16) in OPAMPx_CSR.
3. Set VM_SEL[1:0] to 1x in OPAMPx_CSR (inverting input not externally connected).
4. Clear VP_SEL to 0 in OPAMPx_CSR (GPIO connected to VINP).
5. As soon as OPAEN is set in OPAMPx_CSR, the signal on the OPAMP_VINP pin is amplified by the selected gain, and visible on the OPAMP_VOUT pin.

Note: To avoid saturation, the input voltage must stay below V_{DDA} divided by the selected gain.

Figure 205. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



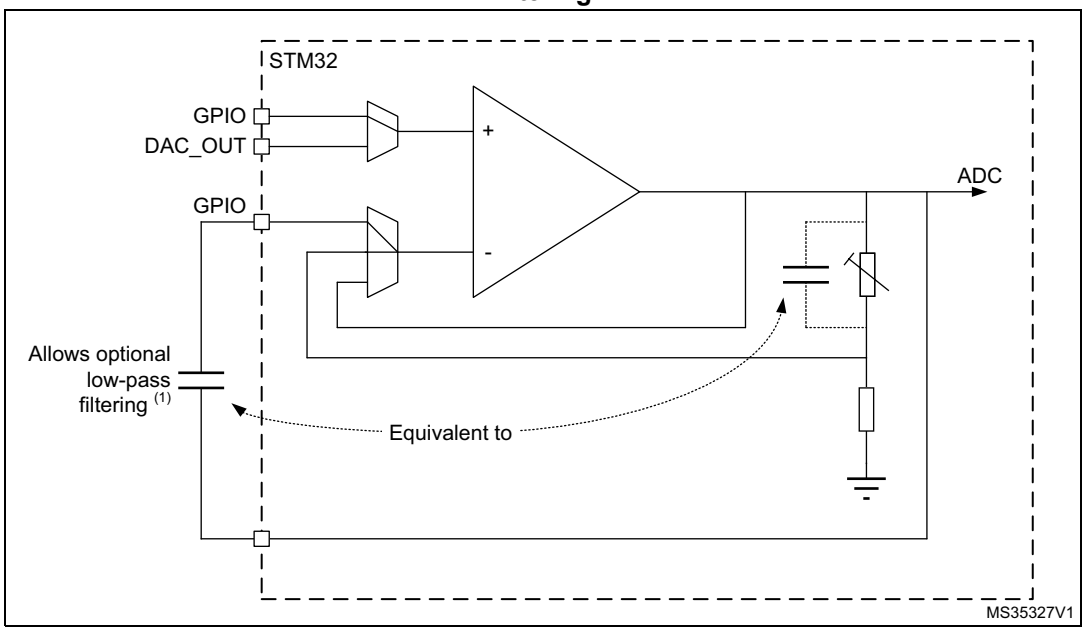
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external set, is detailed below (all bits in OPAMPx_CSR):

1. Configure OPAMODE[1:0] to 10 (internal PGA enabled).
2. Set PGA_GAIN[1:0] to the selected PGA gain (2, 4, 8 or 16).
3. Clear VM_SEL[1:0] to 00 or 01(GPIO connected to VINM).
4. Clear VP_SEL to 0 (GPIO connected to VINP).

Any external connection on VINP can be used in parallel with the internal PGA. For example, a capacitor can be connected between VOUT and VINM for filtering purpose (see datasheet for the value of resistors used in the PGA resistor network).

Figure 206. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cut-off frequency.

26.3.5 Calibration

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each OPAMP offset can be trimmed by the user. Specific registers allow different trimming values for normal and low-power modes.

The calibration purpose is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows the inputs offset voltage to be reduced to less than ± 1.5 mV within stable voltage and temperature conditions.

For each OPAMP and each mode, two trimming values can be trimmed: one for N differential pair and one for P differential pair.

The registers used to trim the offsets for each operational amplifiers are:

- OPAMPx_OTR for normal mode
- OPAMPx_LPOTR for low-power mode

Each register is composed of five bits for P differential pair trimming, and five bits for N differential pair trimming. These are the ‘user’ values.

The user is able to switch from ‘factory’ values to ‘user’ trimmed values, with the USERTRIM bit in OPAMPx_CSR. This bit is reset at startup and the ‘factory’ value is applied by default to the OPAMP trimming registers.

The trimming values can be changed in calibration or in functional mode.

The offset trimming registers are typically configured after the initialization of the calibration operation (CALON set to 1 in OPAMPx_CSR). When CALON = 1, the OPAMP inputs are disconnected from the functional environment.

Setting CALSEL to 1 in OPAMPx_CSR initializes the offset calibration for the P differential pair (low-voltage reference used).

Clearing CALSEL to 0 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the CALOUT bit in OPAMPx_CSR reflects the influence of the trimming value selected by CALSEL and OPALPM. When the value of CALOUT switches between two consecutive trimming values, this means that those two values are the best trimming values. The CALOUT flag needs up to 1 ms after the trimming value is changed to become steady (see $t_{OFFTRIM}$ max delay specification in the “electrical characteristics” section of the datasheet).

Note: The closer the trimming value is to the optimum trimming value, the longer it takes to stabilize (with a maximum stabilization time remaining below 1 ms in any case).

When the calibration operation is done, OPAHSM must be cleared to 0 in OPAMPx_CSR.

Table 218. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	OPALPM	CALON	CALSEL	V_{OUT}	CALOUT
Normal operating mode	1	0	0	X	Analog	0
Low-power mode	1	1	0	X	Analog	0
Power down	0	X	X	X	Z	0
Offset cal high for normal mode	1	0	1	0	Analog	X
Offset cal low for normal mode	1	0	1	1	Analog	X
Offset cal high for low-power mode	1	1	1	0	Analog	X
Offset cal low for low-power mode	1	1	1	1	Analog	X

Calibration procedure

Here are the steps to perform a full calibration of either one of the operational amplifiers:

1. Set OPAEN to 1 in OPAMPx_CSR to enable the OPAMP and set OPA_RANGE = 1 in OPAMP1_CSR.
2. Clear OPAHSM to 0 in OPAMPx_CSR.
3. Set CALON and USERTRIM to 1 in OPAMPx_CSR.
4. Choose a calibration mode (refer to *Table 218*). Steps 4 to 5 must be repeated four times. For the first iteration, select normal mode, offset cal high (N differential pair), with OPALPM = 0 and CALSEL = 0 in OPAMPx_CSR.
5. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 0, until CALOUT changes to 1 in OPAMPx_CSR.

   Between the write to OPAMP_OTR and the read of the CALOUT value, make sure to wait for the $t_{OFFTRIM}$ max delay specified in the “electrical characteristics” section of the datasheet, to get the correct CALOUT value.

   The commutation means that the offset is correctly compensated and that the corresponding trim code must be saved in OPAMP_OTR.

6. Repeat steps 4 to 5 for:
   – Normal mode and offset cal low
   – Low-power mode and offset cal high
   – Low-power mode and offset cal low

   If a mode is not used, the corresponding calibration can be skipped.

All OPAMPs can be calibrated at the same time.

Note: During the whole calibration phase:

- *The external connection of the OPAMP output must not pull up or down currents higher than 500 µA.*
- *OPAMODE[1:0] must be set up to 00 or 01 (PGA disable) or 11 (internal follower).*

26.4 OPAMP low-power modes

Table 219. Effect of low-power modes on the OPAMP

Mode	Description
Sleep	No effect.
Stop0/1	No effect, OPAMP registers content is kept.
Stop2/3	The OPAMP is disabled; the register content is kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby mode.

26.5 OPAMP registers

26.5.1 OPAMP1 control/status register (OPAMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPA_RANGE	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
r	rw	rw	rw		rw	rw				rw	rw	rw	rw	rw	rw

Bit 31 **OPA_RANGE:** OPAMP range setting
This bit must be set before enabling the OPAMP. This bit affects all OPAMP instances.
0: Reserved
1: OPAMP range set

Bit 30 **OPAHSM:** OPAMP high-speed mode
This bit is effective for both normal and low-power modes.
0: Normal mode (standard slew rate)
1: Increased consumption to improve the slew rate

Bits 29:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT:** OPAMP calibration output
During the calibration mode, the offset is trimmed when this signal toggles.

Bit 14 **USERTRIM:** Selection of factory or user offset trimmed values
This bit is active for normal and low-power modes.
0: Factory trim code used
1: User trim code used

Bit 13 **CALSEL:** Calibration selection
0: NMOS calibration (200 mV applied on OPAMP inputs)
1: PMOS calibration (V_{DDA} - 200 mV applied on OPAMP inputs)

Bit 12 **CALON:** Calibration mode enable
0: Normal mode
1: Calibration mode (all switches opened by hardware)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP_SEL:** Noninverted input selection
0: GPIO connected to VINP
1: DAC connected to VINP

Bit 9 **VM_SEL:** Inverting input selection
This bit is used only when OPAMODE[1:0] = 00, 01, or 10.
0: GPIO connected to VINM (valid also in PGA mode for filtering)
1: Inverting input not externally connected

Bits 8:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN[1:0]:** OPAMP programmable amplifier gain value
00: Internal PGA gain 2
01: Internal PGA gain 4
10: Internal PGA gain 8
11: Internal PGA gain 16

Bits 3:2 **OPAMODE[1:0]:** OPAMP PGA mode
00-01: Internal PGA disabled
10: Internal PGA enabled, gain programmed in PGA_GAIN
11: Internal follower

Bit 1 **OPALPM:** OPAMP low-power mode
The OPAMP must be disabled to change this configuration.
0: Normal mode
1: Low-power mode

Bit 0 **OPAEN:** OPAMP enable
0: OPAMP disabled
1: OPAMP enabled

26.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX

XXXX are factory trimmed values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP[4:0]					Res.	Res.	Res.	TRIMOFFSETN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]:** Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]:** Trim for NMOS differential pairs

26.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR)

Address offset: 0x08

Reset value: 0x0000 XXXX

XXXX are factory trimmed values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMLPOFFSETP[4:0]					Res.	Res.	Res.	TRIMLPOFFSETN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMLPOFFSETP[4:0]:** Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]:** Low-power mode trim for NMOS differential pairs

26.5.4 OPAMP2 control/status register (OPAMP2_CRS)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
r	rw	rw	rw		rw	rw				rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OPAHSM**: OPAMP high-speed mode
This bit is effective for both normal and high-speed modes.
0: Normal mode (standard slew rate)
1: Increased consumption to improve the slew rate

Bits 29:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT**: OPAMP calibration output
During calibration mode, the offset is trimmed when this signal toggles.

Bit 14 **USERTRIM**: Selection of factory or user offset trimmed values
This bit is active for normal and low-power modes.
0: Factory trim code used
1: User trim code used

Bit 13 **CALSEL**: Calibration selection
0: NMOS calibration (200 mV applied on OPAMP inputs)
1: PMOS calibration (V_{DDA} - 200 mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enable
0: Normal mode
1: Calibration mode (all switches opened by hardware)

Bit 11 Reserved, must be kept at reset value.

Bit 10 **VP_SEL**: Noninverted input selection
0: GPIO connected to VINP
1: DAC connected to VINP

Bit 9 **VM_SEL:** Inverting input selection
This bit is used only when OPAMODE[1:0] = 00, 01, or 10.
0:GPIO connected to VINM (valid also in PGA mode for filtering)
1: Inverting input not externally connected

Bits 8:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN[1:0]**: OPAMP programmable amplifier gain value
00: Internal PGA gain 2
01: Internal PGA gain 4
10: Internal PGA gain 8
11: Internal PGA gain 16

Bits 3:2 **OPAMODE[1:0]**: OPAMP PGA mode
00-01: internal PGA disabled
10: Internal PGA enabled, gain programmed in PGA_GAIN
11: Internal follower

Bit 1 **OPALPM**: OPAMP low-power mode
The OPAMP must be disabled to change this configuration.
0: Normal mode
1: Low-power mode

Bit 0 **OPAEN**: OPAMP enable
0: OPAMP disabled
1: OPAMP enabled

26.5.5 OPAMP2 offset trimming register in normal mode (OPAMP2_OTR)

Address offset: 0x14

Reset value: 0x0000 XXXX

XXXX are factory trimmed values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMOFFSETP[4:0]					Res.	Res.	Res.	TRIMOFFSETN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMOFFSETP[4:0]**: Trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMOFFSETN[4:0]**: Trim for NMOS differential pairs

26.5.6 OPAMP2 offset trimming register in low-power mode (OPAMP2_LPOTR)

Address offset: 0x18

Reset value: 0x0000 XXXX

XXXX are factory trimmed values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIMLPOFFSETP[4:0]					Res.	Res.	Res.	TRIMLPOFFSETN[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **TRIMLPOFFSETP[4:0]**: Low-power mode trim for PMOS differential pairs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **TRIMLPOFFSETN[4:0]**: Low-power mode trim for NMOS differential pairs

26.5.7 OPAMP register map

Table 220. OPAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	OPAMP1_CSR	OPA_RANGE	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	PGA_GAIN [1:0]		OPAMODE [1:0]		OPALPM	OPAEN
	Reset value	0	0															0	0	0	0		0	0				0	0	0	0	0	0
0x04	OPAMP1_OTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMOFFSETP [4:0]					Res.	Res.	Res.	TRIMOFFSETN [4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X
0x08	OPAMP1_LPOTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLPOFFSETP[4:0]					Res.	Res.	Res.	TRIMLPOFFSETN[4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X
0x10	OPAMP2_CSR	Res.	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL	Res.	Res.	Res.	PGA_GAIN [1:0]		OPAMODE [1:0]		OPALPM	OPAEN
	Reset value		0															0	0	0	0		0	0				0	0	0	0	0	0
0x14	OPAMP2_OTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMOFFSETP [4:0]					Res.	Res.	Res.	TRIMOFFSETN [4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X
0x18	OPAMP2_LPOTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIMLP OFFSETP[4:0]					Res.	Res.	Res.	TRIMLP OFFSETN[4:0]				
	Reset value																				X	X	X	X	X				X	X	X	X	X

Refer to *Section 2.3* for the register boundary addresses.

27 Audio digital filter (ADF)

27.1 ADF introduction

The audio digital filter (ADF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- metering

The ADF features one digital serial interface (SITF0) and one digital filter (DFLT0) with flexible digital processing options in order to offer up to 24-bit final resolution.

The ADF serial interface supports several standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

The ADF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition, it is possible to insert a high-pass filter.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter bypass, filter order, decimation ratio. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A sound activity detector (SAD) is available for the detection of sounds or voice signals. The SAD is connected at the output of the DFLT0. Several parameters can be programmed in order to adjust properly the SAD to the sound environment. The SAD strongly reduces the power consumption by preventing the storage of samples into the system memory, as long as the observed signal does not match the programmed criteria.

The digital processing is performed using only the kernel clock. The ADF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

27.2 ADF main features

- AHB Interface
- 1 serial digital input:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- 1 flexible digital filter path including:
  - A MCIC filter configurable in $Sinc^4$ or $Sinc^5$ filter with an adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple

  - A high-pass filter to cancel the DC offset
  - Gain control
  - Saturation blocks
- Clock absence detector
- Sound activity detector
- 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay the selected bitstream
- One trigger input
- Autonomous functionality in Stop mode(s)
- DMA can be used to read the conversion data
- Interrupts services

27.3 ADF implementation

Table 221. ADF features(1)

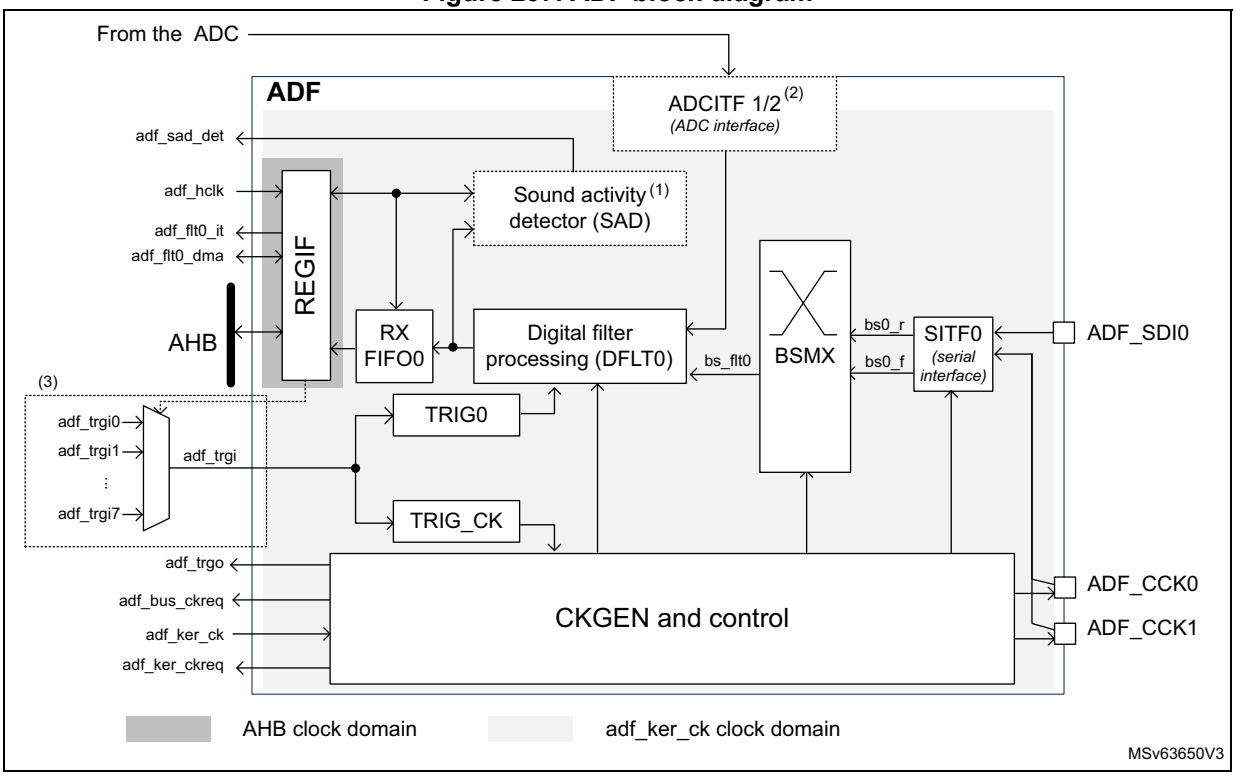
Mode or feature	ADF1
Number of filters (DFLTx) and serial interfaces (SITFx)	1
MDF_CKIy/ADF_CKI0 connected to pins	-
Sound activity detection (SAD)	X
RXFIFO depth (number of 24-bit words)	4
ADC connected to ADCITF1	ADC1
ADC connected to ADCITF2	ADC2
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	-
Main path with CIC4, CIC5	X
Main path with CIC1,2, 3 or FastSinc	-
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X
Autonomous in Stop modes	X(2)

1. ‘X’ = supported, ‘-’ = not supported.
2. Only Stop 0 and Stop 1 modes.

27.4 ADF functional description

27.4.1 ADF block diagram

Figure 207. ADF block diagram



1. Refer to *Section 27.3: ADF implementation* to check if the SAD is available.
2. Refer to *Section 27.3: ADF implementation* to check if the ADCITF is available, and which ADCs are connected.
3. The number of trigger inputs depends on the product. Refer to *Table 224* for details.

27.4.2 ADF pins and internal signals

Table 222. ADF external pins

Pin name	Pin type	Remarks
ADF_SDI0	Input	Data signal from external sensors
ADF_CCKy (y = 0,1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

Table 223. ADF internal signals

Signal name	Signal type	Remarks
adf_trgi	Input	Trigger inputs to control the acquisition (see *Table 224: ADF trigger connections* for details)
adf_trgo	Output	Trigger output for synchronizing with other MDF instances

Table 223. ADF internal signals (continued)

Signal name	Signal type	Remarks
adf_flt0_dma	Input/output	DMA request/acknowledge signals for the ADF processing chain
adf_flt0_it	Output	Global interrupt signals
adf_bus_ckreq	Output	Bus interface clock request output
adf_ker_ckreq	Output	Kernel clock request output
adf_ker_ck	Input	Kernel clock input
adf_hclk	Input	AHB bus interface clock input
adf_sad_det	Output	SAD sound detection: 1 means that detecting sound
adf_adcitf1_dat[15:0]	Input	ADCITF1 data input
adf_adcitf2_dat[15:0]	Input	ADCITF2 data input

Table 224. ADF trigger connections

Trigger name	Direction	Trigger source/destination
adf_trgi0	Input	exti15
adf_trgi1		tim1_trgo
adf_trgi2		tim3_trgo
adf_trgi3		tim6_trgo
adf_trgi4		-
adf_trgo	Output	-
adf_sad_det		-

Table 225. MDF ADC data connections

ADC data bus name	ADC source
mdf_adcitf1_dat[15:0]	adc1_dat
mdf_adcitf2_dat[15:0]	adc2_dat

27.4.3 Serial input interface (SITF)

The SITF0 input interface allows the connection of the external sensor to the digital filter via the bitstream matrix (BSMX). The SITF0 can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The data from the serial interface is routed to the filter in order to perform the PDM to PCM conversion and the sound activity detection.

The serial interface is enabled by setting the SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ΣΔ modulator.

Note: Before enabling the *serial interface, the user must insure that the adf_proc_ck is already enabled (see Section 27.4.5: Clock generator (CKGEN) for details).*

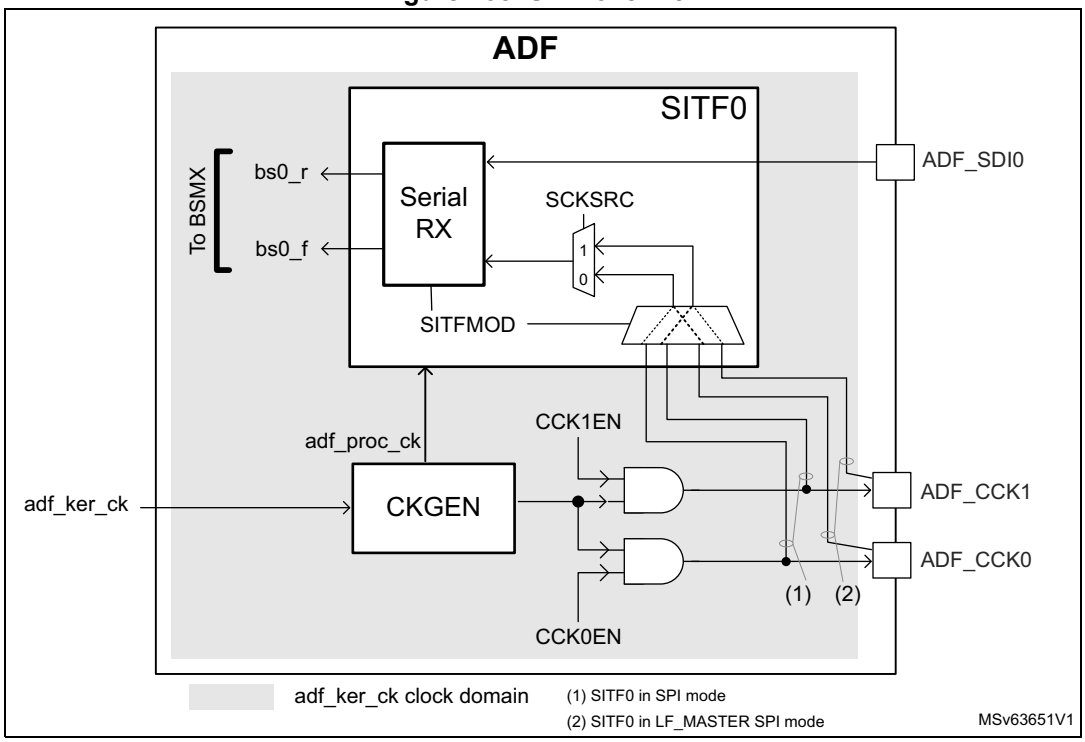
The SITF0 is controlled via the *ADF serial interface control register 0 (ADF_SITF0CR)*.

As shown in the *Figure 208*, ADF_CCK0 or ADF_CCK1 can be selected as clock source, in order to sample the incoming bitstream:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on the ADF_CCK0 or ADF_CCK1 pin.
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source is the clock directly provided by the CCKDIV to the ADF_CCK0 or ADF_CCK1 pin.

See *Table 226* for additional information.

Figure 208. SITF overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the use of an adf_proc_ck clock frequency, only two times bigger than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the ADF must provide the bitstream clock to the external sensors via ADF_CCK0 and ADF_CCK1 pins. The ADF receives the bitstream data via the serial data input ADF_SDI0.

For the SITF0, the application must select the same clock than the one provided to the external sensor (ADF_CCK0 or ADF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the adf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select ADF_CCK0 or ADF_CCK1 clock for the capture of the data received via the ADF_SDI0 pin.

The ADF can generate a clock to the sensors via ADF_CCK0 or ADF_CCK1 if needed.

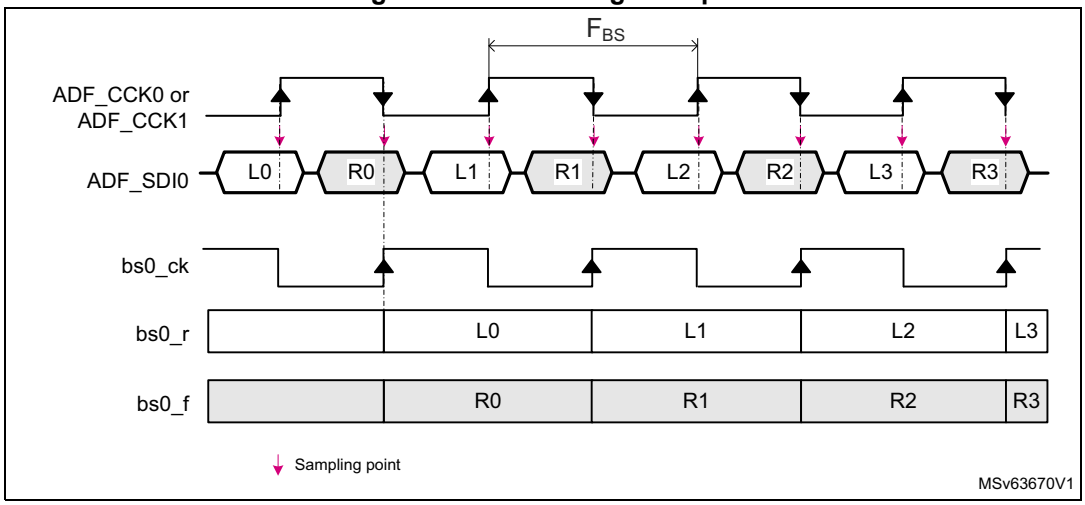
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITF0 always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bs0_f)
- bitstream received using the bitstream clock rising edge (bs0_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 209. SPI timing example



To properly synchronize/receive the data stream, the adf_proc_ck frequency must be adjusted according to the constraints listed in *Table 227*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the SITF0. If the ADF does not detect clock transitions for a duration of STH[4:0] x $T_{adf_proc_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the *ADF serial interface control register 0 (ADF_SITF0CR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF = 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

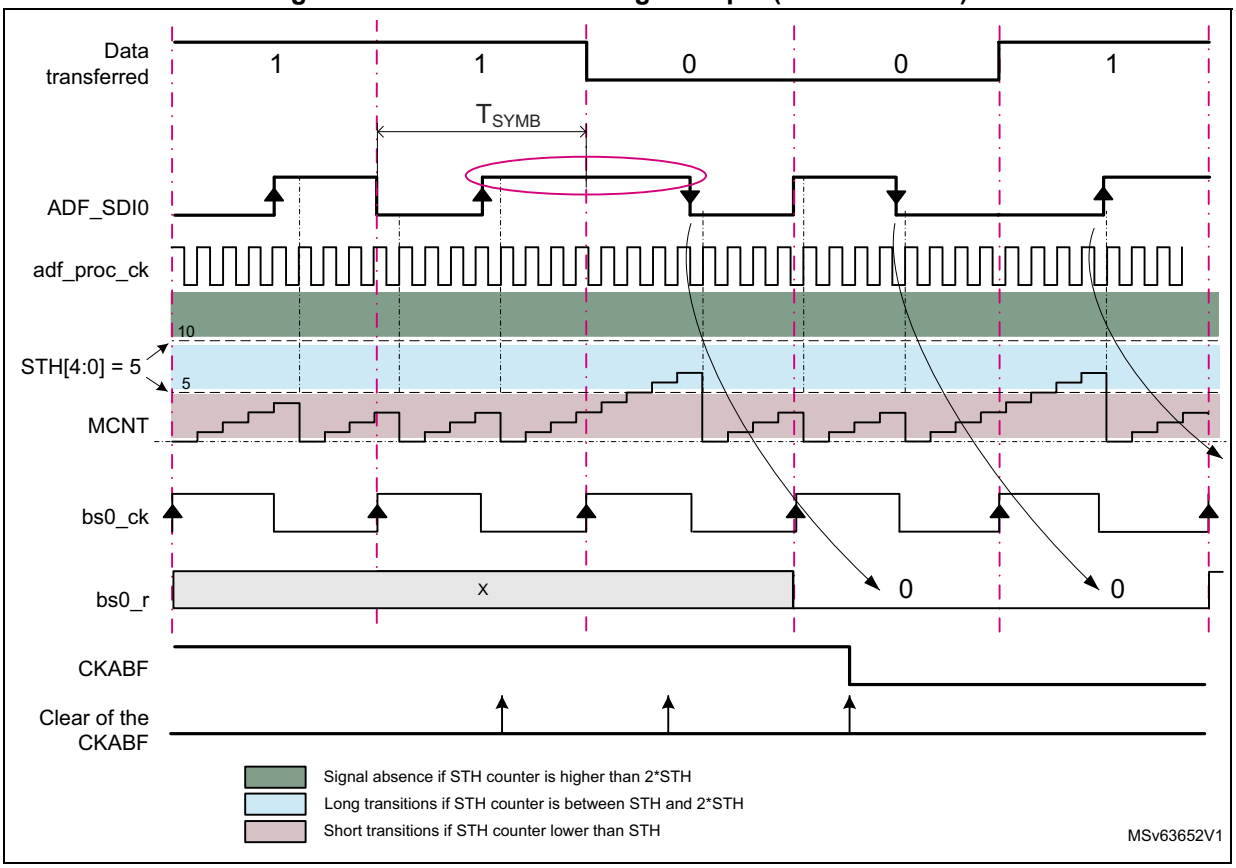
In Manchester coded format, the ADF receives data stream from the external sensor via the ADF_SDI0 pin only.

The ADF_CCK0 and ADF_CCK1 pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bs0_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 210. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program STH[4:0] in the *ADF serial interface control register 0 (ADF_SITF0CR)*. The STH[4:0] bitfield is used by the SITF0 to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the ADF_SDI0 input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. *Figure 210* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1) \times T_{adf_proc_ck} < T_{SYMB} < (2 \times STH \times T_{adf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the adf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the adf_proc_ck clock must be adjusted according to the constraints listed in *Table 227*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{adf_proc_ck}$, or when the SITF0 is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the ADF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 210*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized the hardware immediately set the CKABF flag to 1.
2. Read the CKABF flag:
   – If CKABF= 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the ADF kernel clock frequency ($F_{adf_ker_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}):

1. Provide a valid adf_proc_ck to the SITF0.

   The adf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{adf_proc_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{adf_proc_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = $\text{round}\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{adf_proc_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{adf_proc_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

27.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the ADF. Refer to *Section 27.3* to check the situation for this product.

The ADF allows the connection of up to two ADCs to the filter path. For the filter, the DATSRC[1:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)* allows the application to select data from the ADCs.

Warning: The ADF does not support receiving interleaved data from one of the ADCITF input.

27.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the ADF:

- AHB clock (adf_hclk) used for the register interface
- kernel clock (adf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

These clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the ADF_CCK0 and ADF_CCK1 pins. All those clocks are generated from the adf_ker_ck.

The processing clock (adf_proc_ck) is used to run all the signal processing and to re-sample the incoming serial or parallel stream.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the ADF_CCK0 and ADF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

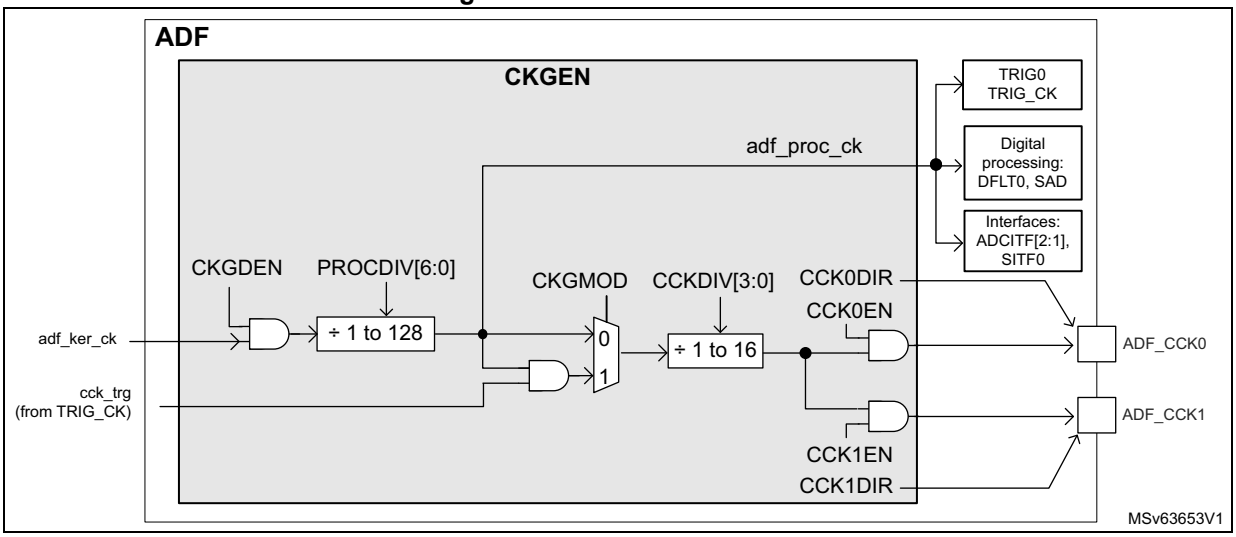
The adf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All the bits and fields controlling the CKGEN are in the *ADF clock generator control register (ADF_CKGCR)*.

Figure 211. CKGEN overview



The trigger logic for CKGEN is handled by the block TRG_CK. As shown in *Figure 216*, the CCKDIV divider can be triggered on the rising or falling edge of an external trigger source. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the ADF_CCKy (y = 0,1) clock of several ADF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the ADF_CCK0 or ADF_CCK1 pin thanks to CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 226. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The ADF_CCKy pin is in input. An external clock can be connected to the ADF_CCKy pin and used by the SITF0 in order to decode the serial stream
0	1	The ADF_CCKy pin is in output. No clock is generated. The ADF_CCKy pin is driven low.
1	1	The ADF_CCKy pin is in output. A clock is generated on the ADF_CCKy pin. The SITF0 can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCyDIR = 0 is not shown must be avoided (no interest).

Note: The adf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITF0 or DFLT0).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0], TRGSENS, CCK1EN and CCK0EN.
4. Set CKGDEN to 1.

When needed, at any moment, CCK[1:0]EN bitfield value can be changed without disabling the clock generator.

Clock frequency constraints

Table 227 shows the frequency constraints to receive and process properly the samples.

Note: The reshape filter (RSFLT) needs up to 24 cycles of adf_proc_ck clock to process one sample.

Table 227. Clock constraints with respect to the incoming stream(1)

SITF0 mode	ADF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{ADF_CCKy} max frequency limited to 5 MHz	
	$F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
MASTER SPI SLAVE SPI	F_{ADF_CKx} max frequency limited to 25 MHz	
	$F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$

1. F_{ADF_CCKy} represents the frequency of clock received via ADF_CCKy, or generated via ADF_CCKy. F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

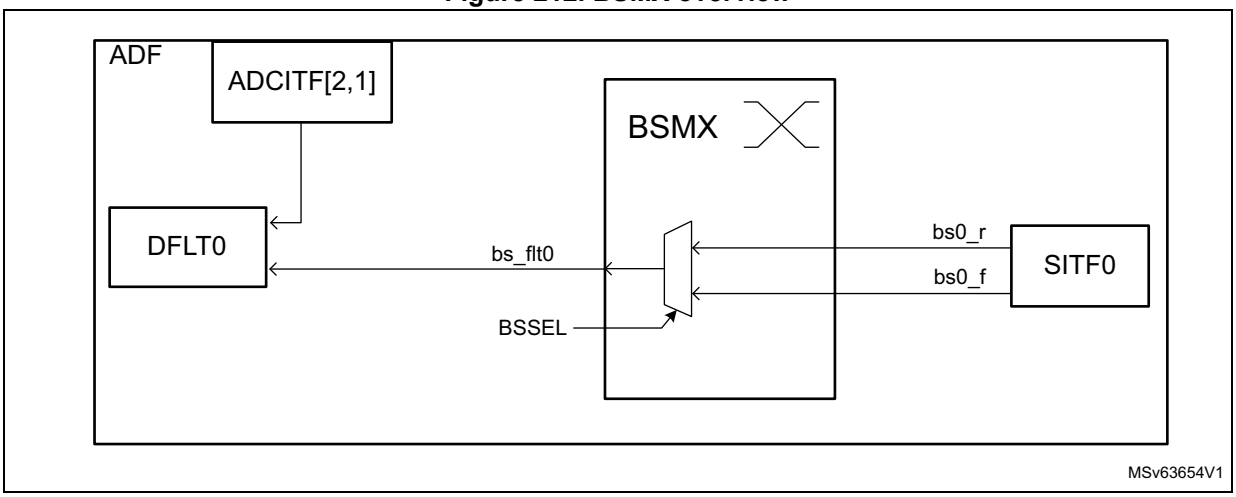
27.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from the serial interface SITF0 and provides the selected stream to the digital filter DFLT0.

As shown in *Figure 207*, the SITF0 provides two bitstreams (bs0_r and bs0_f) to the BSMX.

The application to select the wanted stream via the *ADF bitstream matrix control register 0 (ADF_BSMX0CR)*. This selection is intended to be static.

Figure 212. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the DFLT0 is enabled. The following steps are needed to change the value of BSMX:

1. Set DFLTEN of DFLT0 to 0.
2. Wait for BSMXACTIVE = 0.
3. Program BSSEL[4:0].
4. Set DFLTEN of DFLT0 to 1.

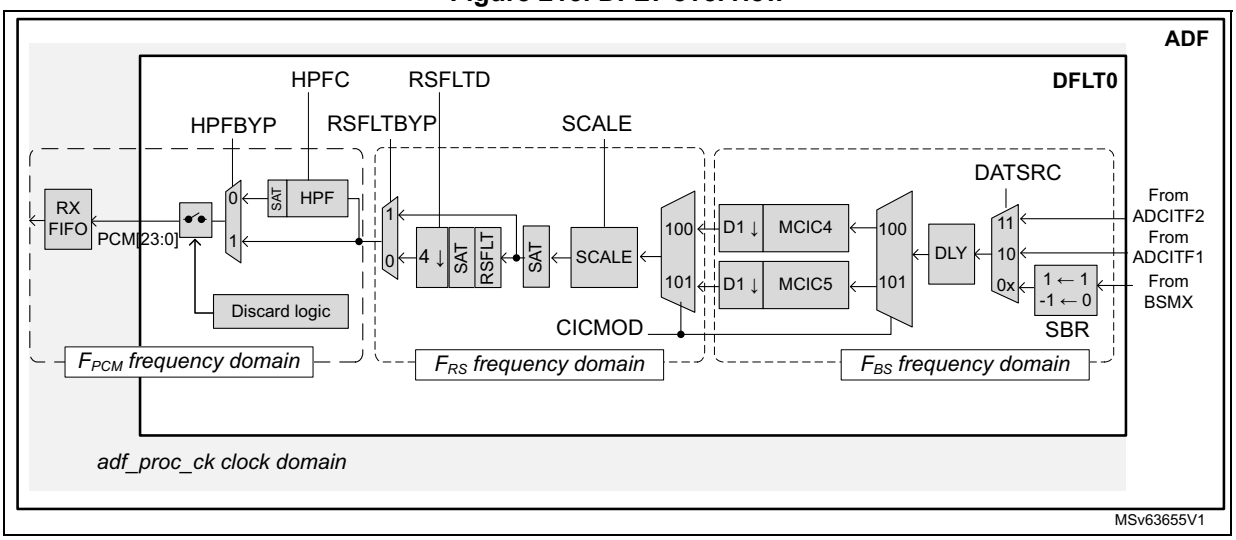
27.4.7 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- clock skipper delay (DLY)
- MCIC decimation filter that can be configured in $Sinc^4$ or $Sinc^5$
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- receive RXFIFO

Figure 213 shows the filter path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 213. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

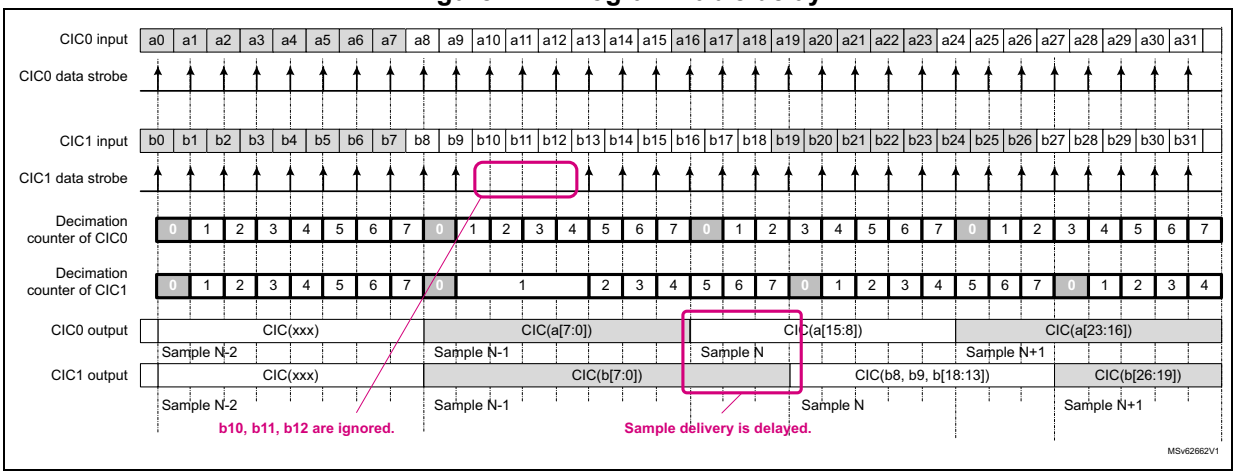
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate must be applied to the incoming stream. This feature can be used when the ADF is synchronized with another MDF instance (if present in the product) for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is done by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

Table 214 shows an example on how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another MDF instance (if present in the product).

Figure 214. Programmable delay



The CIC of the ADF (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is built from input samples a[23:16] while the sample N+1 of CIC1 is built from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.

2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   – If the DFLT0 is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   – If the DFLT0 is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the ADF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the ADF is enabled. If the application performs a D1 delay followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The $Sinc^N$ digital filter embedded into the ADF can be configurable in $Sinc^4$ or $Sinc^5$, according to CICMOD:

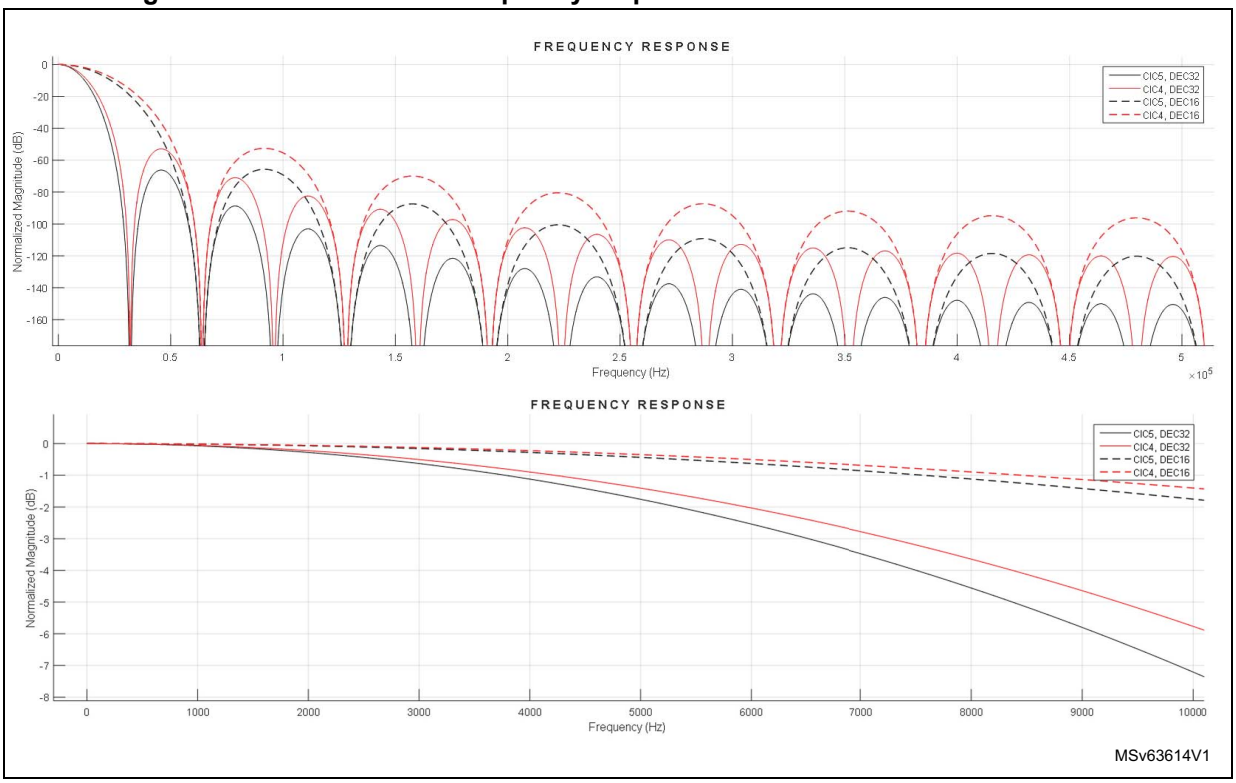
- If CICMOD[2:0] = 4, $Sinc^4$ is selected.
- If CICMOD[2:0] = 5, $Sinc^5$ is selected.

The filters have the following transfer function:

$$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$

where N can be 4 or 5, and D is the decimation ratio. D is equal to MCICD+1.

Figure 215. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}), depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^{N}$$

The decimation ratio can be adjusted from 2 to 512 for the CIC filter.

Table 228 gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream, but can be up to 16 bits when coming from the ADCITF.

Table 228. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITF0)		Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)	
	$Sinc^4$	$Sinc^5$	$Sinc^4$	$Sinc^5$
4	9	11	20	22
8	13	16	24	-
10	15	18	26	-
12	16	19	-	-
16	17	21	-	-
20	19	23	-	-
24	20	24	-	-
32	21	26	-	-
48	24	-	-	-
64	25	-	-	-
76	26	-	-	-

Note: For a full-scale input signal, the decimation ratio must not exceed 76 for a $Sinc^4$ and 32 for a $Sinc^5$.

The LSB parts of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB), and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)*.

SCALE[5:0] can be changed even if the DFLT0 is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of adf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

Table 229 shows the possible gain values.

Table 229. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The ADF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block): The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Gain adjustment policy

To get the best ADF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full-scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF0
– A maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048), for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

The SCALE value in dB must be selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^N}\right)$$

where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

Example: For a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7.

– If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^4}\right) \sim 11 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 11 dB, the closest lower value is - 12dB (SCALE[5:0] = 0x2C).

– If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^4}\right) \sim 23 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

Table 230 proposes gain values for different filter configurations, when the data comes from the SITF0, according to the MCIC order, and the MCIC decimation ratio. This table is not exhaustive and considers a full-scale input signal (see *Section 27.7.5: Total ADF gain* for details).

Table 230. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
8	33.6	51.7	45.7	63.7
12	18.1	39.6	30.1	51.7
16	3.5	27.6	15.6	39.6
20	- 6.0	21.6	6.0	33.6
24	- 12.0	15.6	0	27.6
28	- 20.6	9.5	- 8.5	21.6
32	- 26.6	3.5	- 4.5	15.6
48	-	- 8.5	-	3.5
64	-	- 20.6	-	- 8.5

Reshaping filter (RSFLT)

In addition to the CIC, the ADF offers a reshaping IIR filter mainly dedicated to the audio application, but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

Table 231 shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

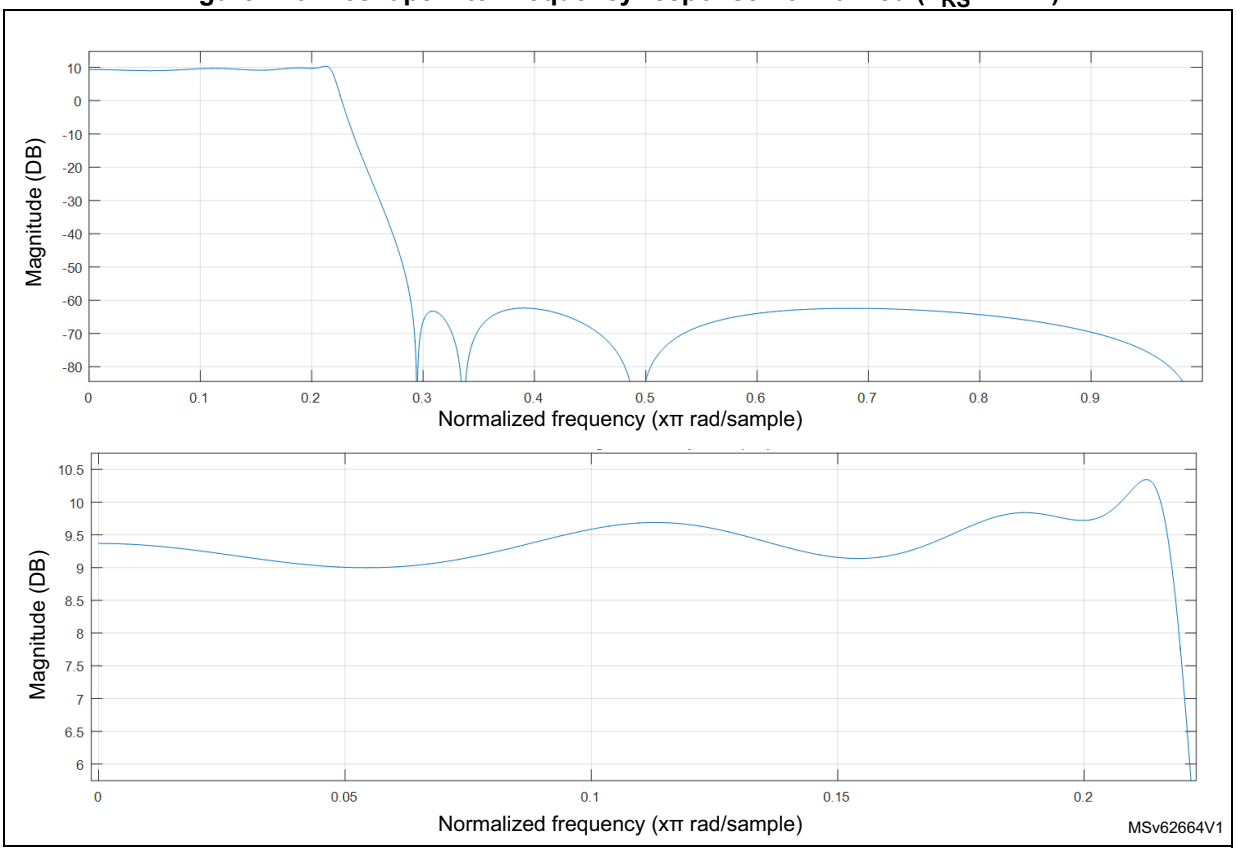
$$F_C = 0.111 \times F_{RS}$$

Table 231. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

Table 216 shows the frequency response of the reshape filter.

Figure 216. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is close to 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of adf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the adf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new samples while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provides a faster adf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source that must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 232. HPF 3 dB cut-off frequency examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

27.4.8 Digital filter acquisition modes

The ADF offers the following modes to perform a data capture:

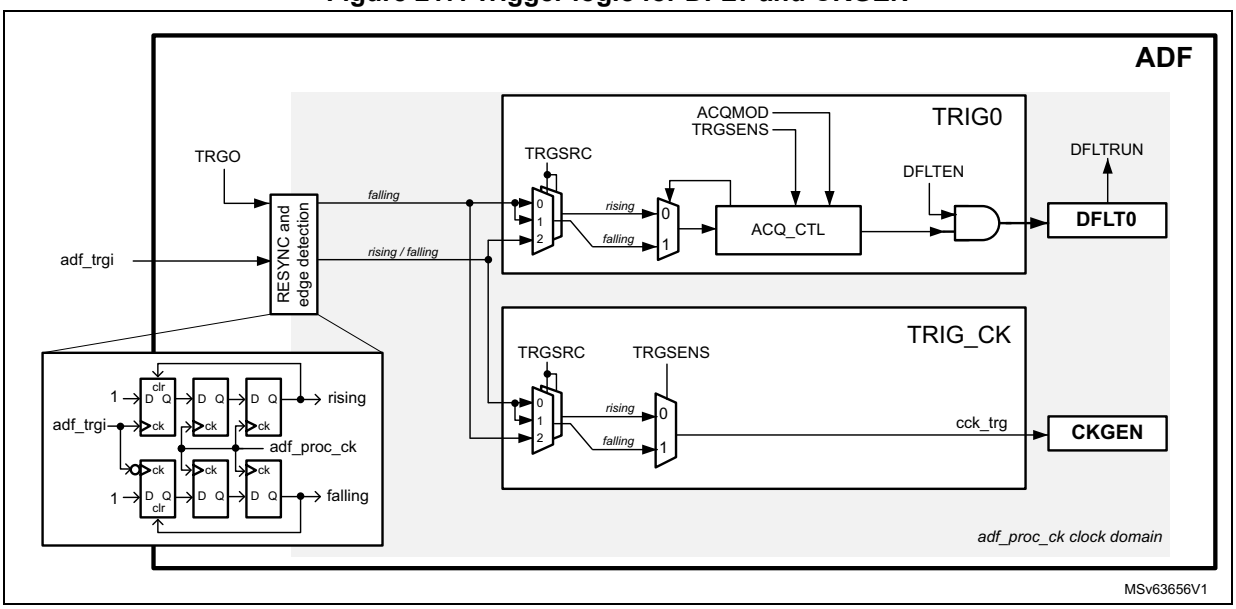
- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode

Note: To perform a data capture, the filter, the interface providing the data (SITF0 or ADCITF) and the CKGEN must be enabled. If needed, the ADF_CCK0 or ADF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This action resets the filter and flushes the RXFIFO. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filter is reset.

Table 217 shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 217. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the re-synchronization of the input trigger to the adf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the adf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select adf_trgi or TRGO bit as trigger sources. The TRGO bit is in the *ADF global control register (ADF_GCR)*.

The edge sensitivity can also be selected.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by simply writing the DFLTEN bit to 1.

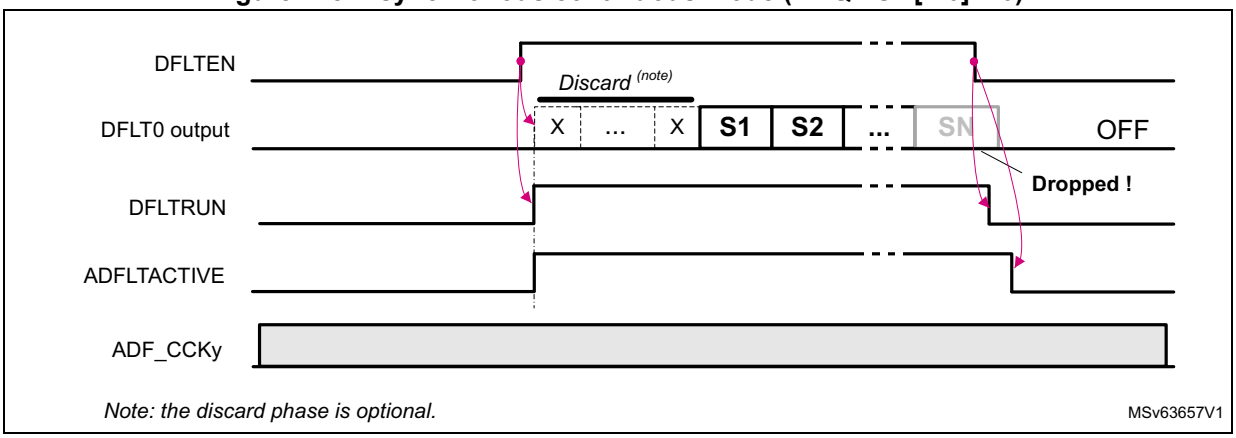
The asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN) so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 236*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the serial data interface.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the acquisition sequence starts immediately.

Figure 218 shows a simplified example of the samples generated by the DFLT0.

Figure 218. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample by simply writing the DFLTEN bit to 1.

The asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 236*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

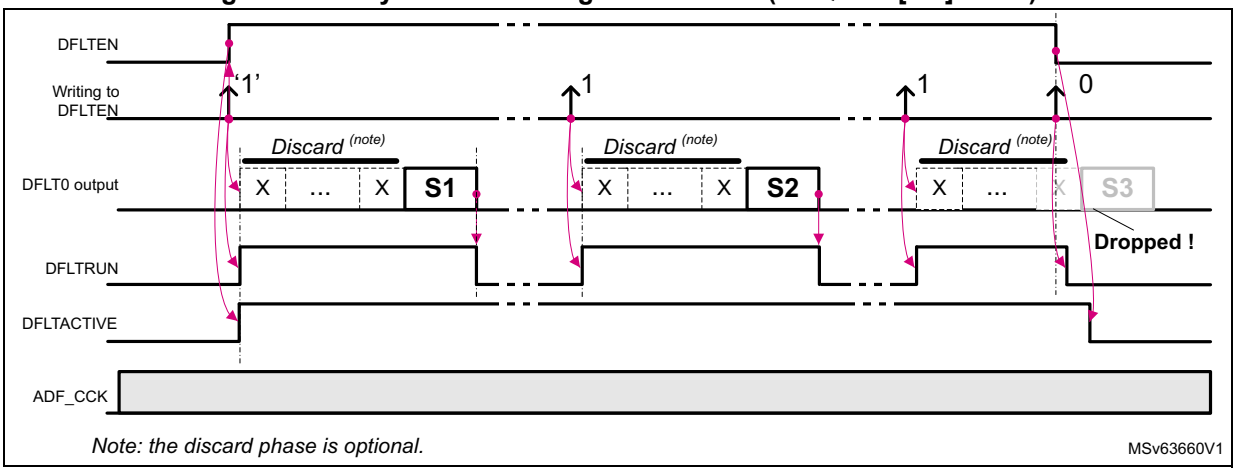
This sequence can be repeated every time a new data must be converted.

As shown in *Figure 218*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 219. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

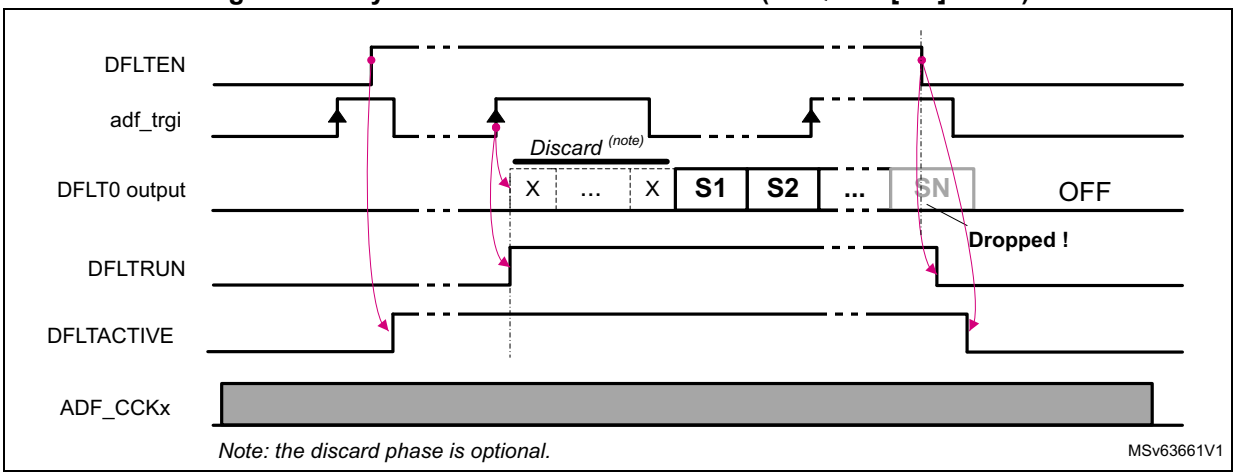
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 236*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 010.
4. Set to 1 the bit SITFEN.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

Figure 220 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. Then the next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 220. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

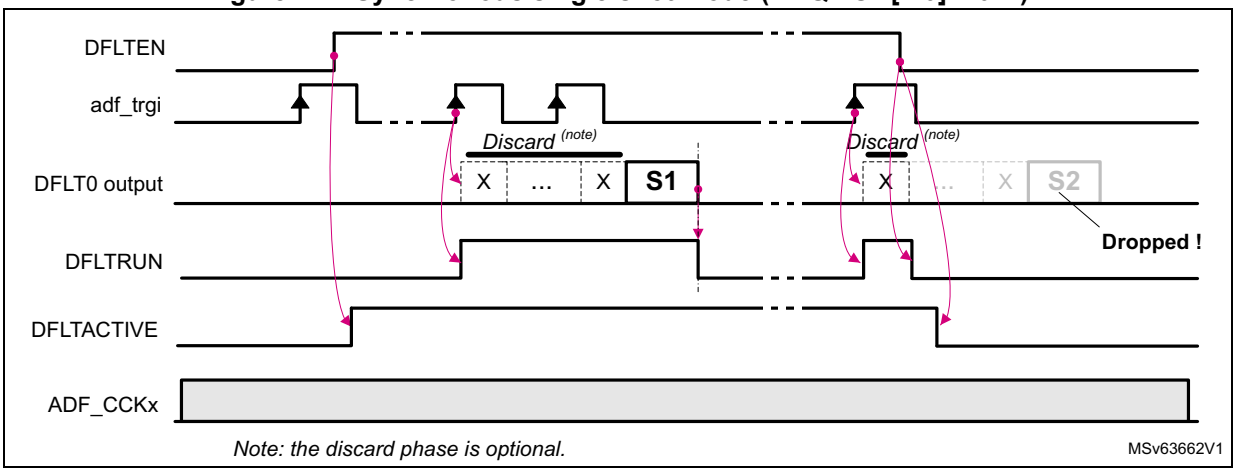
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 236*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition and provides one data to the RXFIFO, then the filter is ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing, or if DFLTEN is set to 0.

Figure 221 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every-time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 221. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 220 shows a case where the DFLTEN is set to 0 while an acquisition is ongoing (the sample S2 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition controlled by consecutive edges of one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

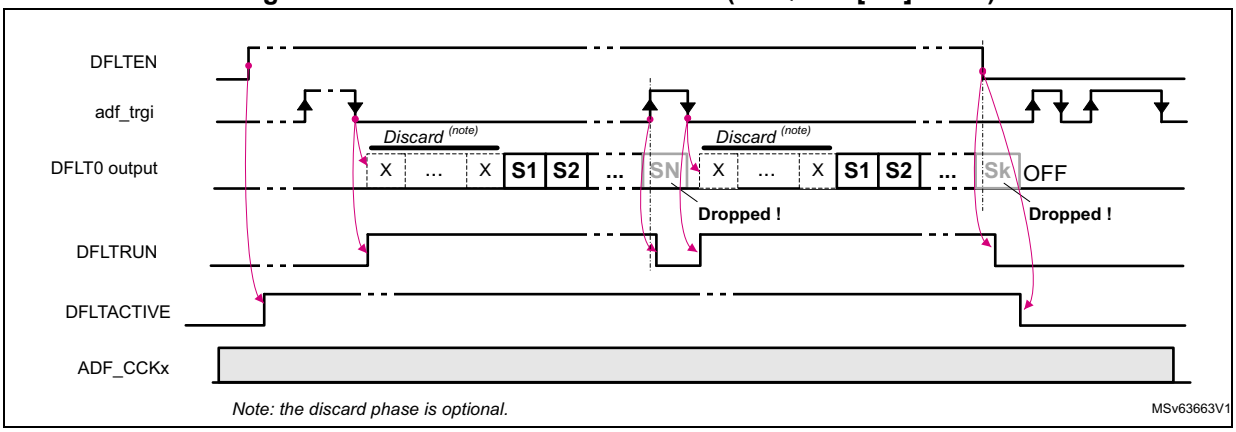
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 236*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 222 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the ADF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the ADF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the ADF waits again for a falling edge on the selected trigger input.

Figure 222. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Starting several filters synchronously

If the ADF is used with MDF instances (if present in the product), it is possible to start simultaneously the acquisition of all the filters. This synchronization capability depends on the way the triggers are connected in the product. Generally, an ADF is able to trigger MDF instances, if its adf_trgo signal is connected as trigger input to those blocks (see *Section 27.4.2: ADF pins and internal signals* to check trigger capabilities).

In the following programming example, one ADF has its adf_trgo signal connected to some MDFs. To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN bits of the filters are set to 0):

On MDFs receiving the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] in order to select the adf_trgo trigger input.
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.

On the ADF generating the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK[1:0] clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. Set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. Set TRGSRC[3:0] to 0 (TRGO selected).
5. Set TRGSENS to 0 (rising edge).
6. Set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot) the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again the bit TRGO to 1.

Discarded samples

The ADF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

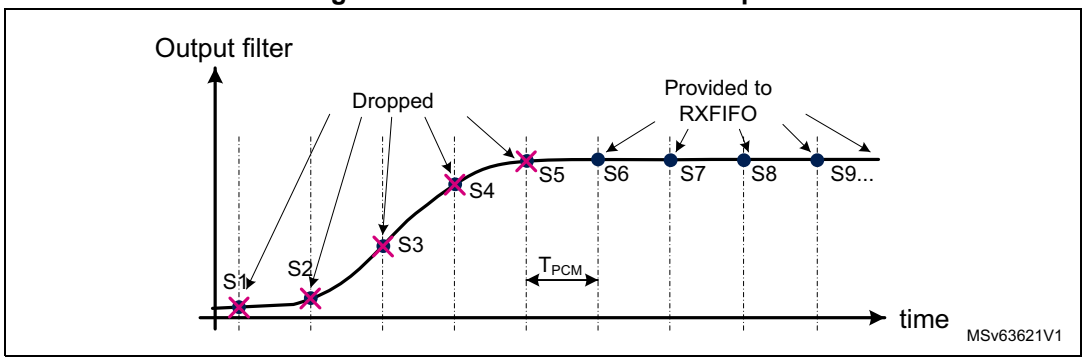
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 217* to *Figure 221*, and *Figure 223*.

In the example shown in *Figure 223*, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO is S6.

Figure 223. Discard function example



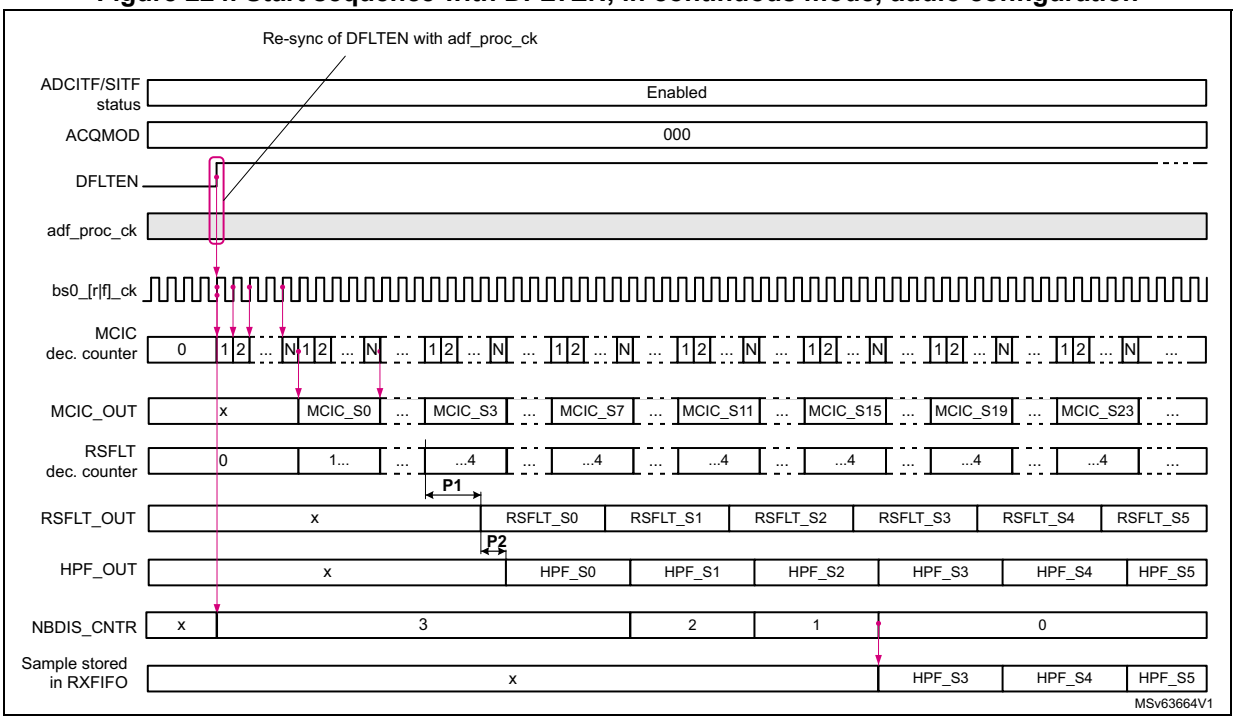
27.4.9 Start-up sequence examples

Figure 224 details a start of acquisition sequence of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLT0 is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITF0 or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a small value to simplify the drawing.

Figure 224. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the ADF processing clock domain. When DFLTEN is detected high, the filter chain is enabled, and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reached its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC, and delivers a sample to the HPF every-time it processes four samples (decimation by 4). The RSFLT needs up to 24 cycles of adf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

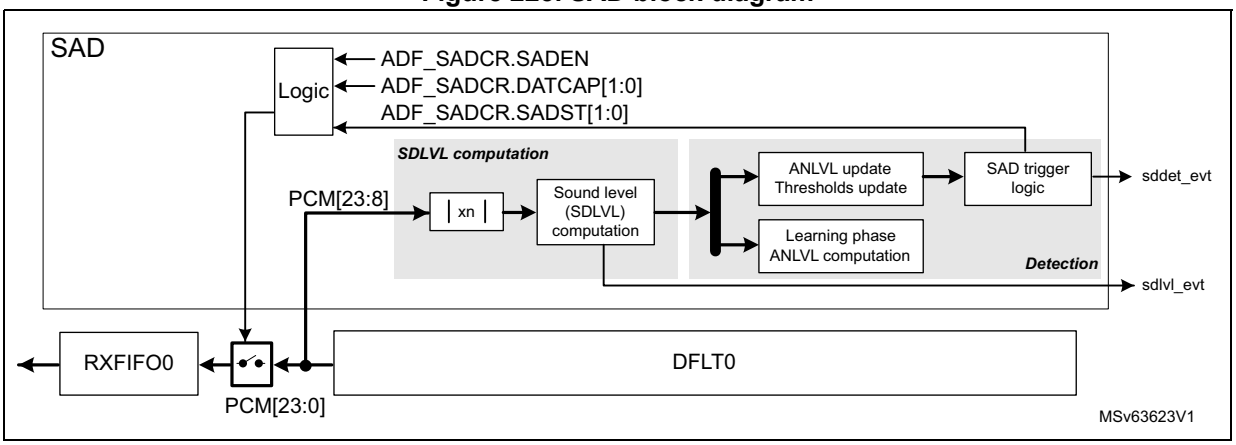
27.4.10 Sound activity detection (SAD)

The SAD is based on the computation of the ambient noise level (ANLVL) and of the short-term sound level (SDLVL). The SAD offers the following ways to detect a sound:

- when the SDLVL reaches a threshold referenced to the ambient noise level
- when the SDLVL reaches a fixed threshold
- when the ANLVL reaches a fixed threshold

As shown in *Figure 225*, the SAD takes the 16 MSB samples from the DFLT0 output.

Figure 225. SAD block diagram



The SAD is highly configurable, and the application can adjust several parameters:

- SAD detection behavior (SADMOD)
- number of samples used to compute the sound level (FRSIZE)
- number of frames used to compute the ambient noise level during the learning phase (LFRNB)
- slope of the ambient noise estimator (ANSLP)
- minimum expected ambient noise level (ANMIN)
- threshold level (SNTHR)
- threshold hysteresis (HYSTEN)
- hangover window in order to filter spurious transitions between DETECT and MONITOR states (HGOVR)
- data capture mode (DATCAP)

SAD detection behavior

The SAD can use the following ways to detect a sound, selected by SADMOD[1:0]:

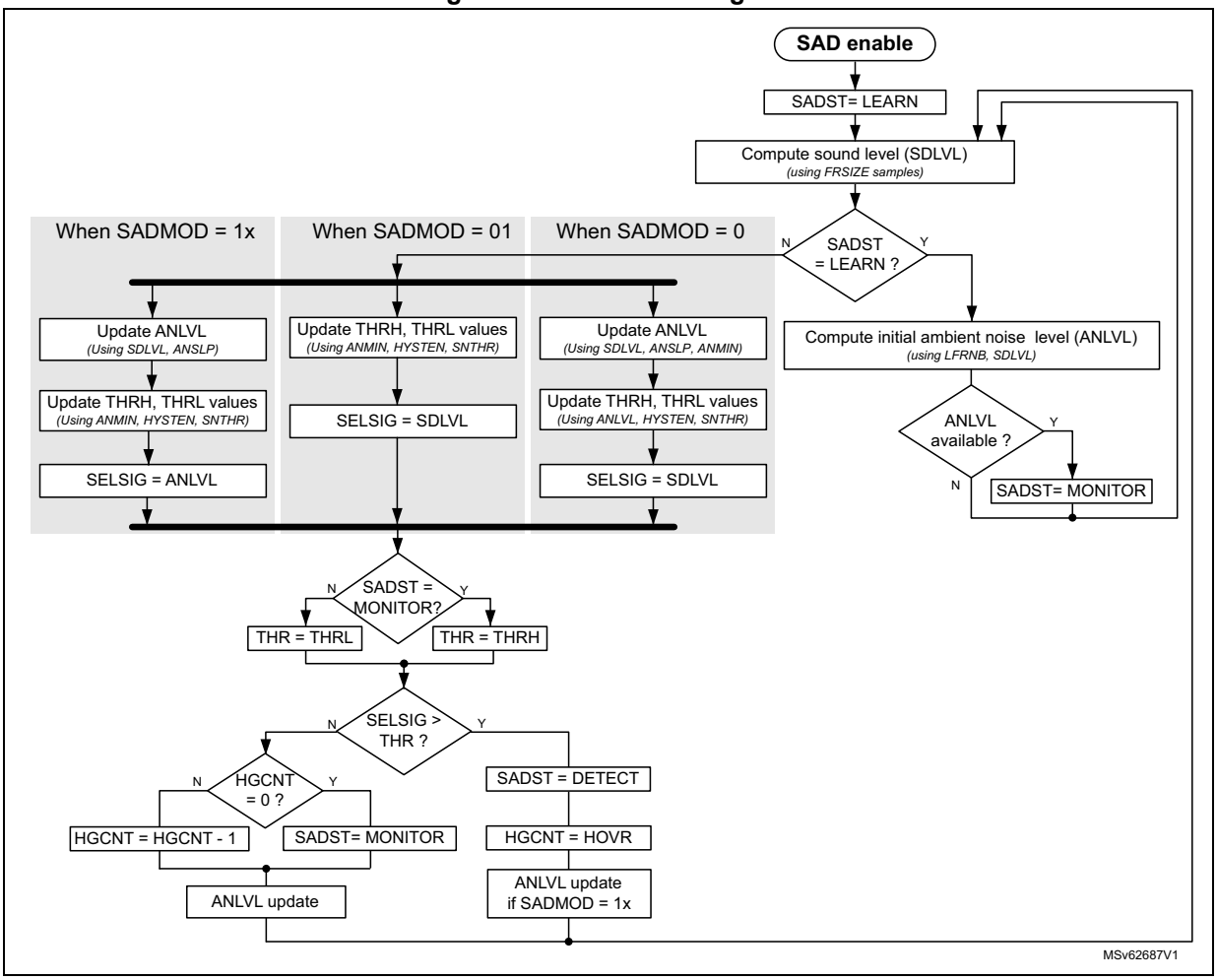
- When SADMOD[1:0] = 0, the SAD works like a voice-activity detection. In this mode, the SAD estimates the ambient noise level according to the computed sound level values. The threshold of the trigger is elaborated from the estimated ambient noise. Finally the current sound level is compared to this threshold. In a first approximation, the SAD triggers if the peak-to-average value of the input signal reaches a level defined by SNTHR[3:0].
- When SADMOD[1:0] = 01, the SAD compares the current sound level (SDLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode allows a fast SAD reaction as the amount of samples used to compute the sound level can be configured via FRSIZE[2:0].
- When SADMOD[1:0] = 1x, the SAD compares the estimated ambient noise level (ANLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode avoids unwanted triggers, due to peak levels, but the SAD reacts more slowly to an input signal variation. It is nevertheless possible to adjust the reaction time via FRSIZE[2:0] and ANSLP[2:0].

SAD states

As shown in *Figure 225*, the SAD works as follows:

1. When enabled (SADEN = 1), the SAD is first in LEARN state to perform a first estimation of the ambient noise level.
2. The SAD continuously computes the short-term sound level (SDLVL) using the samples provided by the DFLT0. The amount of samples used to compute the sound level is given by FRSIZE[2:0]. The samples processed by the DFLT0 can be transferred into the memory or not depending on DATCAP[1:0] value.
3. The initial ambient noise level (ANLVL) is computed using the consecutive sound level values. The application can define how much sound level values are used to perform the computation of this initial ambient noise estimation (LFRNB).
4. When the initial ambient noise level (ANLVL) is computed, the SAD switches to the MONITOR state.
5. Every time a new short-term sound level value is available, the SAD updates the ambient noise level and the thresholds according to the selected detection mode.
6. If the SAD triggers, then the following happens:
   – The SAD switches to DETECT state.
   – The sddet_evt event is asserted.
   – The adf_sad_det signal is set to high.
7. The hangover function insures that the DETECT state is maintained even if the sound level goes below the threshold level for a time given by HGOVR.

Figure 226. SAD flow diagram



Sound level computation (SDLVL)

Once enabled, the SAD computes continuously the sound level value. The sound level represents the average of the absolute value of an amount of PCM samples given by FRSIZE[2:0].

$$SDLVL = \frac{1}{N_{FRSIZE}} \times \sum_{n=1}^{N_{FRSIZE}} |PCM(n)|$$

where N_{FRSIZE} is the amount of PCM samples given by FRSIZE[2:0].

Ambient noise estimation (ANLVL)

The ambient noise level (ANLVL) is computed when SADMOD[1:0] is 00 or 10.

The ambient noise level is computed differently according to the state of the SAD as detailed below:

- ANLVL computation during the LEARN state

  Every time the SAD is enabled, a learning phase is initiated in order to estimate a first value of the ambient noise level. During this phase, the SAD cannot trigger.

  During the LEARN phase, the ambient noise level is computed as follows:

$$ANLVL = \frac{1}{N_{LFRNB}} \times \sum_{n=1}^{N_{LFRNB}} |SDLVL(n)|$$

  where N_{LFRNB} is the amount of frames given by LFRNB[2:0] bitfield.

- ANLVL computation during the MONITOR or DETECT state

  When the learning phase is completed, the SAD updates the ambient noise level in the following way:

  – The SAD computes the new possible values for the ambient noise level:

    ANLVL_UP = ANLVL * (1+2^(ANSLP-12))

    ANLVL_DN = ANLVL * (1-2^(ANSLP-10))

  – The ANLVL takes the ANLVL_DN value if the current sound level is lower than ANLVL_DN, otherwise ANLVL takes the value of ANLVL_UP.

    The ANLVL is not updated if the current sound level is higher than the threshold level, except if SADMOD[1:0] = 10.

  – When SADMOD[1:0] = 0, if the new ANLVL value is lower than ANMIN[12:0], ANLVL is replaced by ANMIN.

  The slope of the noise estimator can be adjusted to optimize the detection of the wanted signal. This slope is adjusted via ANSLP[2:0] in the *ADF SAD configuration register (ADF_SADCFGR)*.

Table 233 shows the allowed values according to the frame size and the sampling rate of the data observed by the SAD. The recommended values when the SADMOD[1:0] = 0 are the ones into the gray shaded cells.

Table 233. ANSLP values versus FRSIZE and sampling rates

FRSIZE	ANSLP values for Fs = 8 kHz			ANSLP values for Fs = 16 kHz		
	Slow(1)	typical(2)	Fast(3)	Slow(1)	typical(2)	Fast(3)
0 (8 samples)	0	1	2	-	0	1
1 (16 samples)	1	2	3	0	1	2
2 (32 samples)	2	3	4	1	2	3
3 (64 samples)	3	4	5	2	3	4
4 (128 samples)	4	5	6	3	4	5
5 (256 samples)	5	6	7	4	5	6
6 (512 samples)	6	7	-	5	6	7

1. The slow slope is equal to - 8.5 dB/s for the negative slope and + 2.1 dB/s for the positive slope.
2. The typical slope is equal to - 17.1 dB/s for the negative slope and + 4.2 dB/s for the positive slope.
3. The fast slope is equal to - 34.2 dB/s for the negative slope and + 8.5 dB/s for the positive slope.

The slopes can also be computed using the following formulas:

$$SLC_{UP} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 + 2^{(ANSLP - 12)})$$

$$SLC_{DN} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 - 2^{(ANSLP - 10)})$$

where F_S is the sampling rate of the stream observed by the SAD and F_{SIZE} is the frame size defined by FRSIZE[2:0].

Threshold computation

The way the threshold value is computed depends on SADMOD[1:0]:

- If SADMOD[1:0] = 0, THRH is obtained by multiplying the current ANLVL value with the gain defined in SNTHR[3:0].

  $$THR_H = ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).
- If SADMOD[1:0] = 01, THRH is obtained by multiplying the current ANMIN[12:0] with the gain defined by SNTHR[3:0].

  $$THR_H = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).

- If SADMOD[1:0] = 1x, THRH is obtained by multiplying the current ANMIN[12:0] by 4.

$$THR_H = ANMIN \times 4$$

This threshold value is then compared to:

$$ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

The hysteresis mode can be enabled to reduce the spurious transitions between MONITOR and DETECT states. In hysteresis mode (HYSTEN = 1), the following threshold values are used:

- THRH when the SAD is in MONITOR state.
- THRL when the SAD is in DETECT state.

Table 234 shows the thresholds values according to SNTHR.

Table 234. Threshold values according SNTHR(1)

SNTHR[3:0]	THR_H		THR_L		Comments
0	LVL + 3.5 dB	LVL x 1.5	LVL + 3.5 dB	LVL x 1.5	No hysteresis
1	LVL + 6.0 dB	LVL x 2	LVL + 3.5 dB	LVL x 1.5	Hysteresis of 2.5 dB
2	LVL + 9.5 dB	LVL x 3	LVL + 6.0 dB	LVL x 2	Hysteresis of 3.5 dB
3	LVL + 12.0 dB	LVL x 4	LVL + 9.5 dB	LVL x 3	Hysteresis of 2.5 dB
4	LVL + 15.6 dB	LVL x 6	LVL + 12.0 dB	LVL x 4	Hysteresis of 3.5 dB
5	LVL + 18.1 dB	LVL x 8	LVL + 15.6 dB	LVL x 6	Hysteresis of 2.5 dB
6	LVL + 21.6 dB	LVL x 12	LVL + 18.1 dB	LVL x 8	Hysteresis of 3.5 dB
7	LVL + 24.1 dB	LVL x 16	LVL + 21.6 dB	LVL x 12	Hysteresis of 2.5 dB
8	LVL + 27.6 dB	LVL x 24	LVL + 24.1dB	LVL x 16	Hysteresis of 3.5 dB
9	LVL + 30.1 dB	LVL x 32	LVL + 27.6dB	LVL x 24	Hysteresis of 2.5 dB

1. LVL must be replaced by ANLVL when SADMOD[1:0] = 0 and by ANMIN for other SADMOD[1:0] values.

When the hysteresis function is disabled, the SAD always use THRH.

Note: The hysteresis mode must not be used when SADMOD[1:0] = 1x.

Trigger logic

The signal compared to this threshold depends also on SADMOD[1:0].

The trigger condition is reached when the selected signal (SELSIG) is bigger than the threshold level.

If the trigger condition is met, the following happens:

- The SAD switches to DETECT state.
- The SAD refreshes the hangover counter with HGOVR.
- The sddet_evt event is asserted if the SAD transits from MONITOR to DETECT.

- The adf_sad_det signal is set to high.

The SAD remains in DETECT state as long as the trigger condition is met or the hangover down-counter is different from 0.

The sddet_evt event indicates when the SAD enters and/or exits the DETECT state. This event is used to generate an interrupt when a sound is detected or when a sound is no longer detected:

- When DETCFG = 0, the application receives an event only when the SAD enters the DETECT state.
- When DETCFG = 1, the application receives an event when the SAD enters or exits the DETECT state.

The adf_sad_det signal remains high as long as the SAD is in DETECT state.

The SAD also provides a flag indicating that a new sound level value is available (SDLVLF). The last computed sound level (SDLVL[14:0]) is available in the *ADF SAD sound level register (ADF_SADSDLVR)*, and the last computed ambient noise level (ANLVL[14:0]), in the *ADF SAD ambient noise level register (ADF_SADANLVR)*.

Note: The SAD can work even when the AHB clock is not present. In that case, the SAD does not update SDLVL[14:0] and ANLVL[14:0].

To get the latest valid SDLVL[14:0] and ANLVL[14:0] values, the application must read the ADF_SADSDLVR, and ADF_SADANLVR registers, when the SDLVLF flag goes high. This can be done in the following ways:

- by polling the SDLVLF flag:
  a) Clear the SDLVLF flag by writing SDLVLF to 1.
  b) Wait for SDLVLF = 1, by reading ADF_DFLTxISR.
  c) Read ADF_SADSDLVR and ADF_SADANLVR.
  d) Clear SDLVLF by writing it to 1.
  e) Go to step 2 if other values must to be read.
- by generating an interrupt:
  a) Read ADF_DFLTxISR.
  b) If SDLVLF = 1, read ADF_SADSDLVR and ADF_SADANLVR, and clear SDLVLF by writing it to 1.
  c) Handle other status flags and exit from ISR.

Sample transfer to memory

The SAD offers the following options to control the samples transfer from DFLT0 to the system memory:

- If DATCAP[1:0] = 1x, the samples are transferred into the system memory as soon as DFLT0 and SAD are enabled. The transfer does not depend on the SAD state.
- If DATCAP[1:0] = 01, the samples are transferred into the system memory when the SAD detects a sound (when the SAD is in DETECT state), assuming that DFLT0 and SAD are enabled.
- If DATCAP[1:0] = 0, the samples are not transferred into the memory. This mode can be used if the application only wants to observe but does not need samples for other processing.

Note: DATCAP[1:0] is taken into account only when the SADEN = 1. For example, if the SAD configuration is DATCAP[1:0] = 0, SADEN = DFLTEN = 1, and if the application sets now SADEN to 0, the samples provided by the DFLT0 are transferred to the RXFIFO.

Programming recommendations

To make the SAD function working properly, the ADF must be programmed as follows:

1. Provide the proper kernel clock (adf_ker_ck) to the ADF.
2. Configure the CKGEN and enable it.
3. Configure the SITF and enable it (note that microphones have a settling time of several milliseconds).
4. Configure the DFLT0. A typical setting is the following:
   - CIC5 with a decimation ratio of 12, 16 or 24
   - RSFLT with a decimation ratio of 4
   - HPF with HPFC = 2 or 3

   For a very-low power implementation, the RSFLT can be bypassed.
5. Set SADEN to 0.
6. Wait for SADACTIVE = 0. If SADEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
7. Configure the SAD as follows:
   - Set DATCAP[1:0] to 0, if the application does not want to store the samples into the system memory.
   - Set DATCAP[1:0] to 01, if the application wants to store the samples into the system memory only when the SAD detects a sound.
   - Set DATCAP[1:0] to 11, if the application wants to store the samples into the system memory continuously.
8. Configure the DMA (optional).
9. Enable the SAD.
10. Enable the DFLT0.

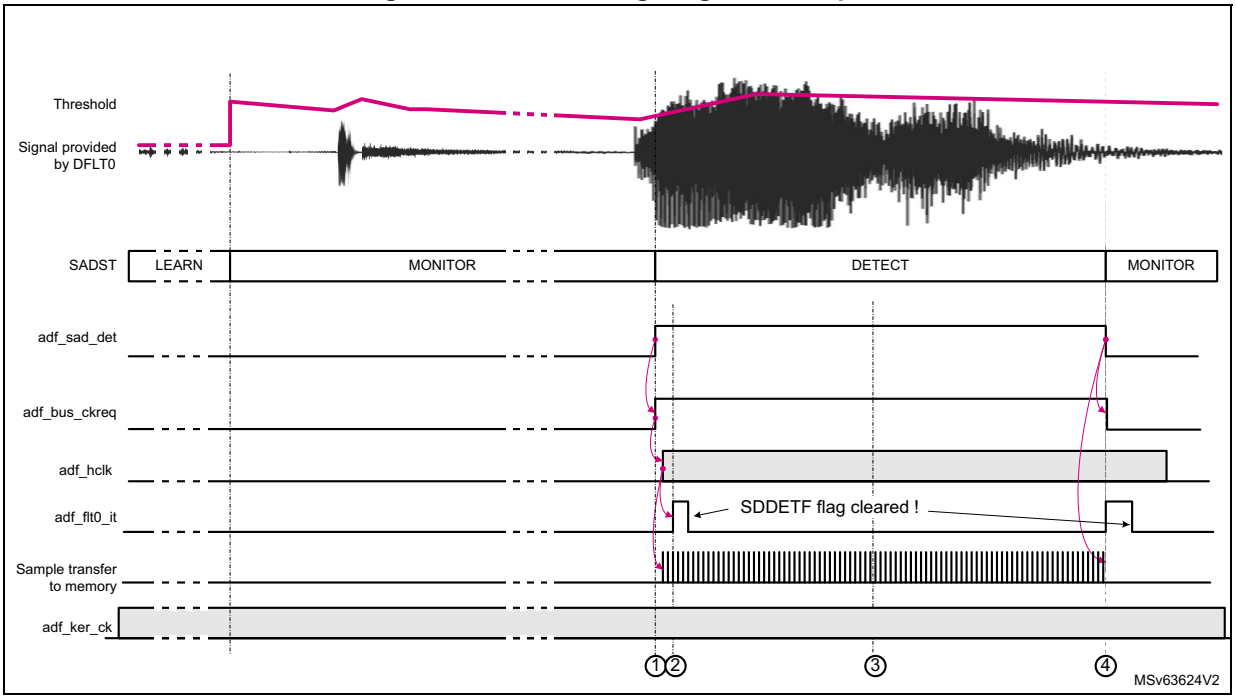
Figure 226 shows a simplified timing diagram when the SAD works with DATCAP[1:0] = 01.

Thanks to the kernel clock (adf_ker_ck), the SAD continuously monitors the audio signal provided by the DFLT0. The threshold is also continuously updated according to the ambient noise level estimation.

- When the SAD detects a sound higher than the programmed threshold (1), the ADF requests the bus clock (adf_bus_ckreq asserted).
- When the bus clock is available (see 2 in *Figure 226*) then:
  - The data transfer to the memory is triggered.
  - The event interrupt (adf_evt_it) can be generated.
- In this example, the event interrupt (adf_evt_it) is used to wake up the application. The interrupt line is released by clearing SDDETF by writing 1 to it.
- As long as the SAD remains in DETECT state, the application waits to get enough samples and calls, for example the keyword recognition algorithm (see 3 in *Figure 226*).
- In the case shown in *Figure 227*, the SAD state (SADST) goes back to MONITOR before the keyword is recognized. If DETCFG is set to 1, an event signals when the SAD goes back to MONITOR state. The SAD stops the transfer of samples into the

memory and the application can clean up the receive buffer for the next detection (see 4 in *Figure 226*).

Figure 227. SAD timing diagram example



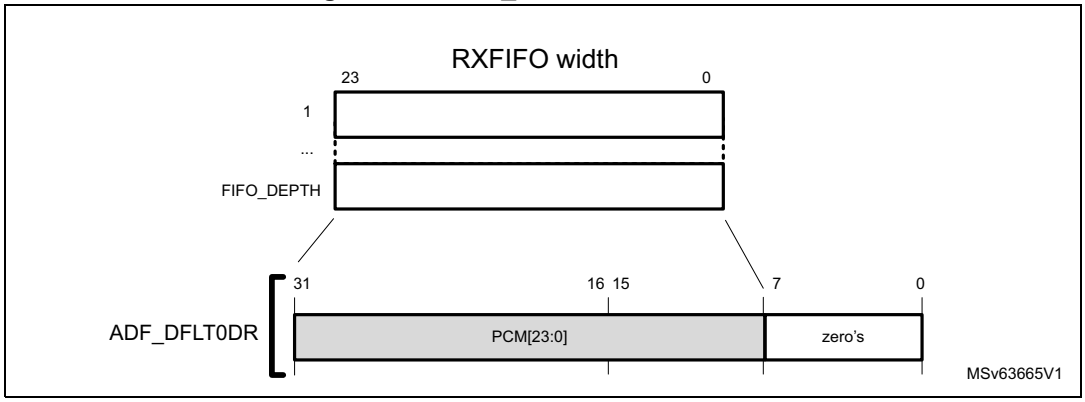
27.4.11 Data transfer to memory

Data format

The samples processed by DFLT0 are stored into a RXFIFO. The application can read the samples stored into these FIFOs via the *ADF digital filter data register 0 (ADF_DFLT0DR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The ADF provides 24-bit left-aligned data. Performing a 16-bit access to ADF_DFLT0DR allows the application to get the 16 most significant bits. Performing a 32-bit access to ADF_DFLT0DR allows the application to get a 24-bit data size.

Figure 228. ADF_DFLTxDR data format



Data re-synchronization

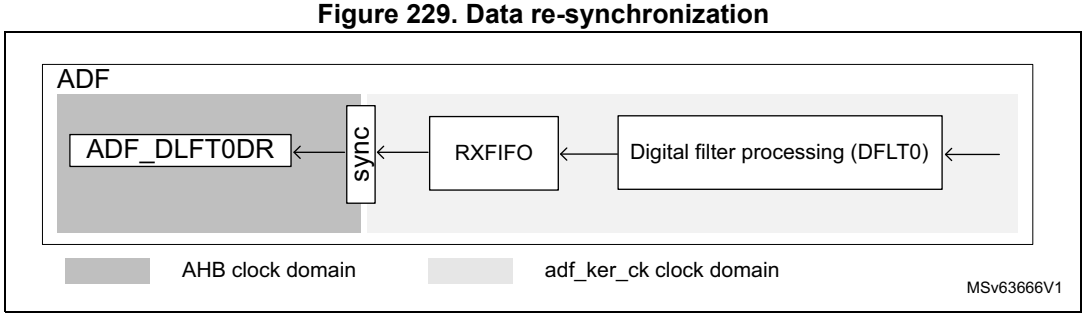
The samples stored into the RXFIFO can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFO is located into the adf_ker_ck clock domain, while ADF_DFLT0DR *is located into the adf_hclk (AHB) clock domain.*

When the AHB clock is available, if ADF_DFLT0DR is empty and if a sample is available into the RXFIFO, this sample is transferred into ADF_DFLT0DR.

The sample transfer from the RXFIFO to ADF_DFLT0DR takes two periods of the AHB clock (adf_hclk) and two periods of the adf_ker_ck clock. The ADF inserts automatically wait-states if the application performs a read operation of ADF_DFLT0DR while the transfer of the new sample from the RXFIFO to ADF_DFLT0DR is not yet completed.

Figure 229. Data re-synchronization



Data transfer

The content of the RXFIFO can be transfered to the memory either by using a DMA channel or interrupt services.

Both single and burst, DMA transfers are supported by the ADF, but the application has to care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data re-synchronization explained in the previous section.

Note: The burst mode is not available in all products (see the DMA section to check if the product supports it).

In addition, the application can select the RXFIFO threshold (FTH bit) in order to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth/2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the RXFIFO is not empty.

The DMA mode of the RXFIFO is enabled via the DMAEN bit in ADF_DFLT0DR.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as ADF_DFLT0DR is read.
- If FTH = 1, as soon as the RXFIFO reaches the threshold level and a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as one data is read. FTHF is set again if the threshold condition is

met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE/2 data.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full, and a new sample from the DFLT0 must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 230 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

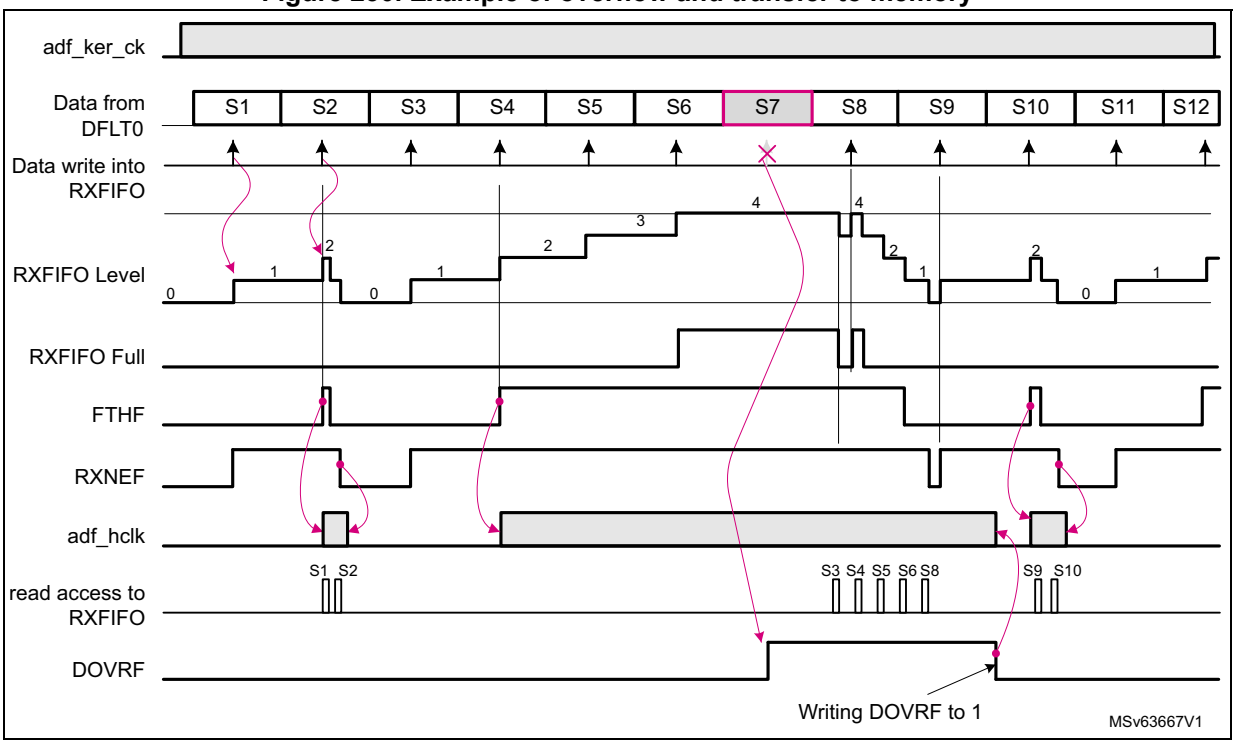
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO, and the ADF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the adf_hclk line, the gray boxes indicate that the ADF requested the AHB clock. *Figure 230* shows the AHB clock available only when the ADF requests it. In real applications, the AHB clock may also be present if the ADF does not request it.

Figure 230. Example of overflow and transfer to memory



27.4.12 Autonomous mode

The ADF can work even if the AHB bus clock is not available (Stop modes). The ADF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the ADF receives a kernel clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the ADF receives the AHB clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The ADF requests the AHB clock in the following situations:
  - when the ADF must transfer data into memory via the DMA

    The data is directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the ADF needs to generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt flag is still active.

27.4.13 Register protection

The ADF embeds some hardware protection to prevent invalid situations.
Table 235 shows the list of write-protected and unprotected fields.

Table 235. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF global control register (ADF_GCR)	TRGO	-	DFLTACTIVE0 = 1
ADF clock generator control register (ADF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
ADF serial interface control register 0 (ADF_SITF0CR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
ADF bitstream matrix control register 0 (ADF_BSMX0CR)	-	BSSEL[4:0]	DFLTACTIVEx = 1
ADF digital filter control register 0 (ADF_DFLT0CR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
ADF digital filer configuration register 0 (ADF_DFLT0CICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)	-	All fields	
ADF delay control register 0 (ADF_DLY0CR)	-	SKPDLY[6:0]	SKPBF = 1

Table 235. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF DFLT0 interrupt enable register (ADF_DFLT0IER)	All fields	-	-
ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)	All fields	-	-
ADF SAD control register (ADF_SADCR)	SADEN	FRSIZE[2:0], DETCFG, SADMOD[1:0], HYSTEN, DATCAP[1:0]	SADACTIVE = 1
ADF SAD configuration register (ADF_SADCFGR)	-	ANMIN[12:0], SNTHR[3:0], HGOVR[3:0], LFRNB[2:0], ANSLP[2:0]	SADACTIVE = 1

All the ADF processing is performed in the adf_proc_ck clock domain. For that reason, enabling or disabling an ADF sub-block may take some time due to the re-synchronization between the AHB clock domain and the adf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for details.

27.5 ADF low-power modes

Table 236. Effect of low-power modes on ADF

Mode	Description
Sleep	No effect. ADF interrupts cause the device to exit the Sleep mode.
Stop(1)	The ADF registers content is kept. Thanks to the kernel clock request feature, if the ADF is clocked by an internal oscillator available in Stop mode, the ADF remains active. The interrupts cause the device to exit Stop mode.
Standby	The ADF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 27.3: ADF implementation* for details about Stop modes supported by the ADF.

27.6 ADF interrupts

To increase the CPU performance, the ADF offers an interrupt line (adf_flt0_it), sensitive to several events.

Note: The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Figure 231. ADF interrupt interface

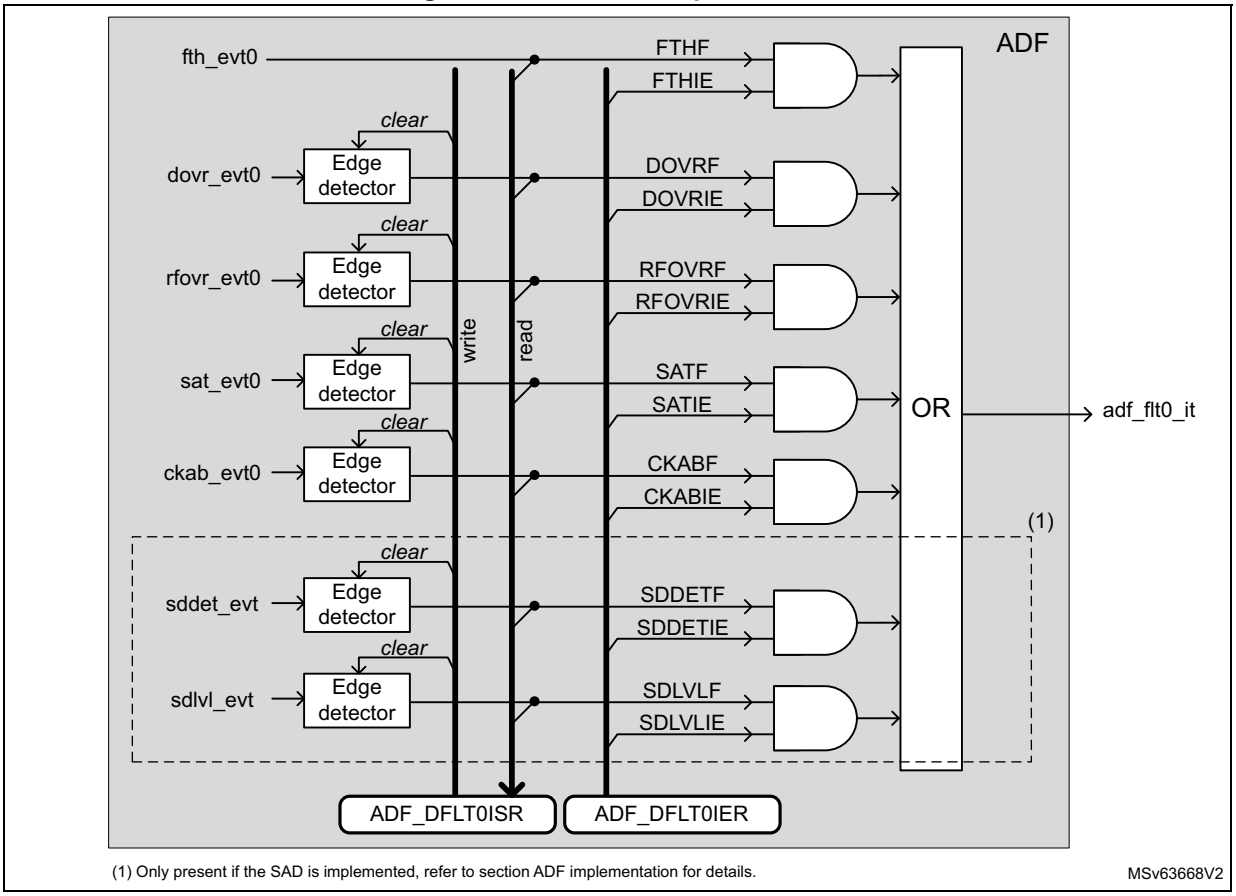


Table 237 shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 237. ADF interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes(1)	Exit Standby mode
ADF_FLT0(2)	RXFIFO threshold reached	FTHF	Read ADF_DFLT0DR until RXFIFO level is lower than the threshold.	Yes	Yes	No
	RXFIFO overrun	DOVRF	Write DOVRF to 1.			
	RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
	Saturation detection	SATF	Write SATF to 1.			
	Channel clock absence detection	CKABF	Write CKABF to 1.			
	SAD: sound detected	SDDETF	Write SDDETF to 1.			
	SAD: sound level value available	SDLVLF	Write SDLVLF to 1.			

1. Refer to *Section 27.3: ADF implementation* for details.
2. ADF_FLT0 vector corresponds to the assertion of adf_flt0_it signal.

27.7 ADF application information

27.7.1 ADF configuration examples for audio capture

Table 238 gives some examples of the ADF settings for the digital microphones, focusing on 16 and 48 kHz output data rate.

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (adf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple, and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 238. Examples of ADF settings for microphone capture

Configuration	adf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	adf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{ADF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

27.7.2 Programming examples

Example 1

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using only the CIC4, with a decimation of 48, assuming that the kernel clock is 1.536 MHz. Typically, this configuration can be used to detect sound using the SAD.

Table 239. Programming sequence (CIC4)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 1.536 MHz coming from a RC oscillator.
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0001 0023	PROCDIV = 0 (bypass): adf_proc_ck frequency is 1.536 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 768 kHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Filters configuration (CIC): ADF_DFLT0CICR = 0x0040 2F40	SCALE = 0x04 (12 dB): RSFLT is not enabled. Note that the gain is 8.5 dB, higher than recommended in order to improve signal accuracy for the SAD. Saturation is not an issue in this case, as only the detection of a signal much lower than the full scale is needed. MCICD = 0x2F (decimation by 48) CICMOD = 4 to select a $Sinc^4$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0301	HPFC = 3: cut-off frequency of 16kHz * 0.0095 = 152 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 1: RSFLT bypassed
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = 0x0000 1000	Enable the interrupt events the application wants to handle. In this example, SDDETIE is set to 1 to have an interrupt if a sound is detected.

Table 239. Programming sequence (CIC4) (continued)

Operations	Comments
SAD setting: ADF_SADCR = 0x0000 1400	SADMOD = 1: Trigger based on sound level values FRSIZE = 4: Sound level computed on 128 samples HYSTEN = 0: No hysteresis DETCFG = 0: interrupt generated when the SAD enters in DETECT state DATCAP = 0: samples not transferred to memory SADEN = 0: SAD not yet enabled
SAD setting: ADF_SADCFGR = 0x0025 0001	ANMIN = 0x25: Sound threshold of 37 * 10^SNTHR/20 = 74 LSB(1) HGOVR = 0: four frames (4 x 128 = 512 samples) LFRNB = 0: Learning phase set to two frames. Not useful in this mode. ANSLP = 0: Not used in this mode SNTHR = 1: Threshold set to 6 dB (see ANMIN)
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Enable the SAD: ADF_SADCR = 0x0000 1401 Wait for SADACTIVE flag = 1	Enable first the SAD, in order to be sure that data transfer to memory are blocked by the SAD.
Enable the filter: ADF_DFLT0CR = 0x0000 0001	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 0: DMA interface not used DFLTEN = 1: Filter enabled

1. SNTHR and ANMIN values are computed using the same approach than in *Threshold programming with SADMOD = 01* (detection of a sound higher than 63 dBSPL).

Example 2

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using the CIC5, and the RSFLT, with a total decimation of 64.

Table 240. Programming sequence (CIC5)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 6.144 MHz
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0201 0023	PROCDIV = 2 (division by 3): adf_proc_ck frequency is 6.144 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 1.024 MHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.

Table 240. Programming sequence (CIC5) (continued)

Operations	Comments
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SIPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Digital filter control: ADF_DFLT0CR = 0x0000 0006	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 1: DMA interface not used DFLTEN = 0: Filter disabled FTH = 1: RXFIFO half-full
Filters configuration (CIC): ADF_DFLT0CICR = 0x0010 0F50	SCALE = 0x01(+3.5 dB): Output limited to 22 bits due to RSFLT. MCICD = 0xF (decimation by 16) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0200	HPFC = 2: cut-off frequency of 16 kHz * 0.0025 = 40 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 0: RSFLT not bypassed RSFLTD = 0: Decimation by 4
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = x	Enable the interrupt events the application wants to handle.
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside the ADF_DLT0DR register every time a DMA request is generated.
Start acquisition: ADF_DFLT0CR = 0x0000 0007	The ADF starts to filter data.

27.7.3 Connection examples

Figure 231 shows simple connection examples of the ADF to external sensors.

- Picture on the left: two digital microphones connected to the ADF

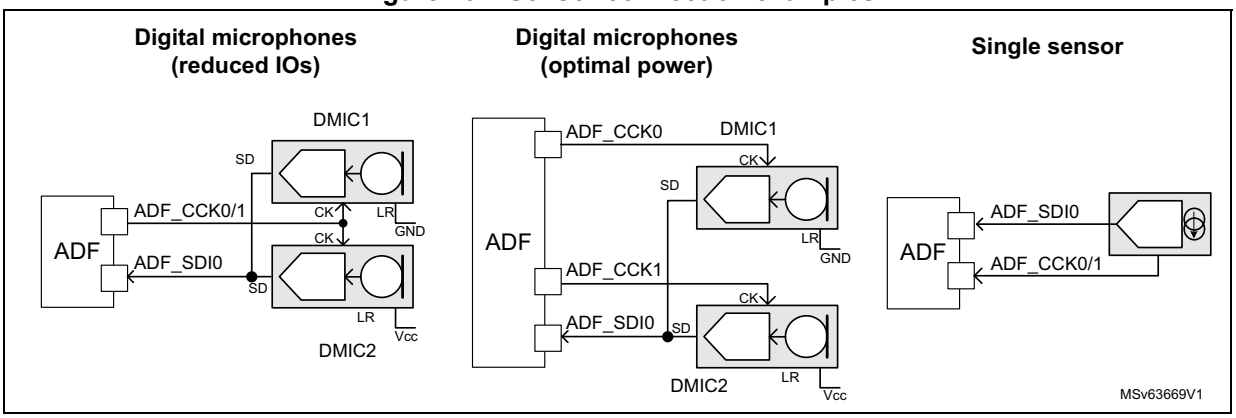
  In this connection, the amount of connections is optimized; DMIC1 and DMIC2 are sharing the same data line, and the same clock line. BSMX allows the application to connect the digital filter either to DMIC1 or to DMIC2. In this configuration when one of the microphone is used, the other is activated as well, as they share the same clock.
- Picture in the center: two digital microphones connected to the ADF

  In this connection, DMIC1 and DMIC2 are sharing the same data line, but have a dedicated clock line. BSMX allows the application to connect the filter either to DMIC1 or to DMIC2. When the application wants to use a microphone, it is possible to keep the other in low-power mode by forcing its clock line to 0 (CCKyEN = 0, CCKyDIR = 1).

- Picture on the right: single sensor connected to the ADF

  It is also possible to configure the CCK0 and CCK1 pins to input in order to connect sensors providing the clock.

Figure 232. Sensor connection examples



27.7.4 Global frequency response

Figure 232 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 40 Hz

Figure 233 shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 233. Global frequency response

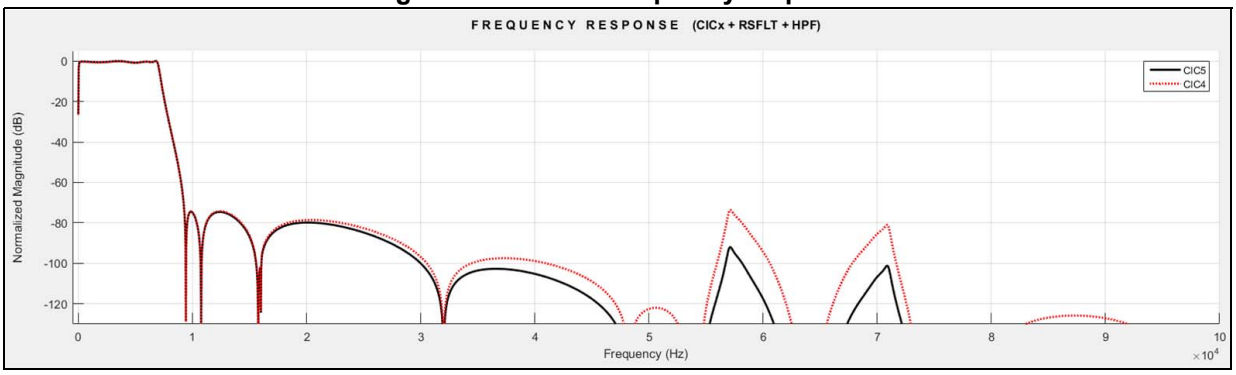


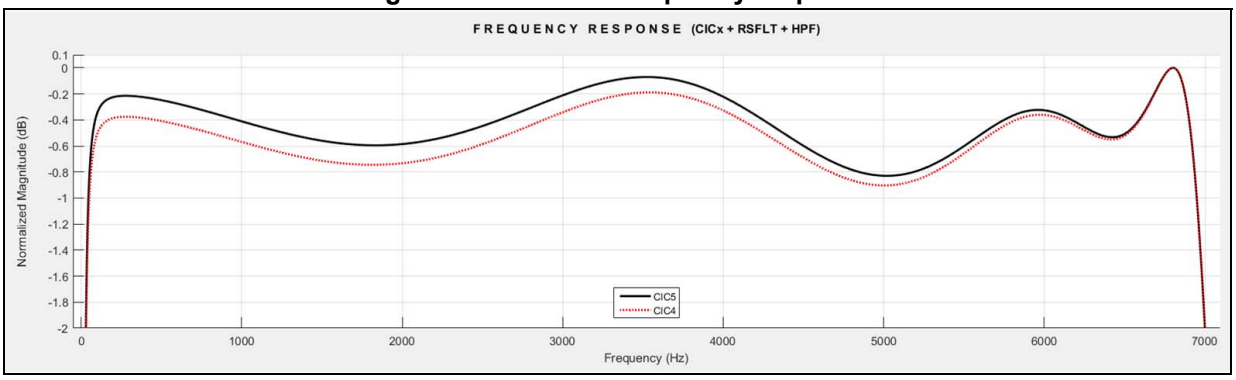
Figure 234 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 dB for CIC4.

The - 3 dB cut-off frequency is 7061 Hz.

Figure 234. Detailed frequency response



27.7.5 Total ADF gain

This section details how to compute the signal level provided by the ADF according to the filter settings.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a − 6 dBFS level (6 dB below full scale).

For example, for the ADF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DSin$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD[8:0]).

DSin represents the data size (in bits) of the signal at CIC input.

Warning: **DS_{CIC} is very important for CIC filter. In order to work fine, DS_{CIC} must not exceed 26 bits.**

The CIC gain G_{CIC} is given by:

$$G_{CIC} = (D1)^N$$

which gives in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Data size at SCALE output

The data size at SCALE output (including the CIC gain), is a key information as the RSFLT starts to have some saturations, if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted.

The signal amplitude at SCALE output is:

$$Asout_{SCALE} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$$

GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

$Asout_{SCALE}$ is the signal amplitude at SCALE output (in LSB), and $Asin_{DFLT}$ is the signal amplitude at CIC input (LSB).

$$DS_{SCALE} = \frac{\ln(Asout_{SCALE})}{\ln(2)} + 1$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.41 dB must be considered.

$$G_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 \text{ typical}$$

Note: The HPF filter has a gain of 0 dB.

SAD gain

The SAD is using only the 16 MSB on the signal, as a consequence, from the SAD point of view, the truncation from 24 to 16 bits can be seen as an attenuation.

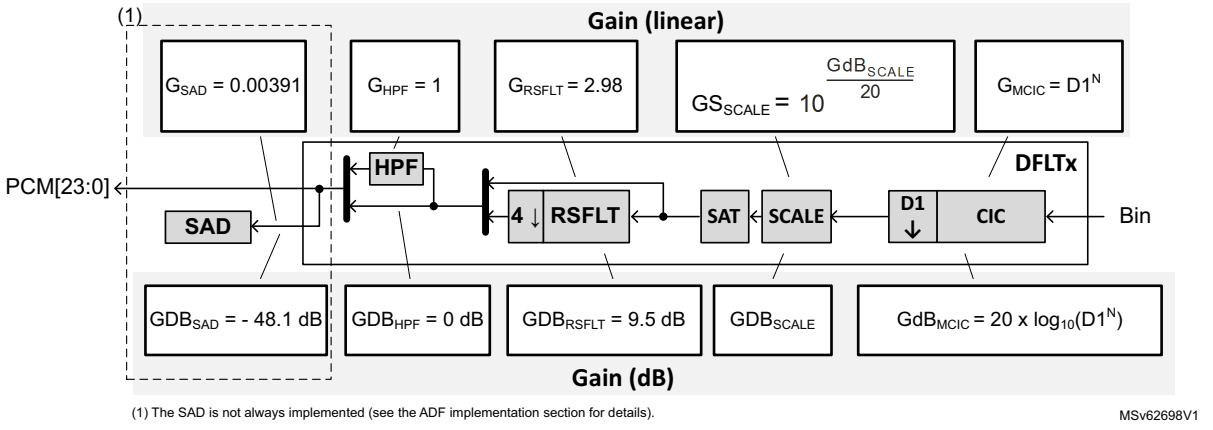
$GdB_{SAD} = -48.1dB$

and

$G_{SAD} = 0.003906$

The *Figure 235* shows a simplified view of the filter path and gives, for each significant component, the expression of the bit growth and the gain.

Figure 235. Simplified DFLT view with gain information



Note: The SAD is not always implemented (see Section 27.3 for details).

Table 241 summarizes of the final data size for different filter configurations.

Table 241. Output signal levels

Filter configurations	Final signal amplitude (LSB)
CIC + RSFLT + HPF + SAD	Samples provided to the RXFIFO: $Asout_{RXFIFO} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ Samples provided to the SAD: $Asout_{SAD} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times 0.003906 \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits
CIC + RSFLT (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits
CIC (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 24 bits

Example using the main filter chain

If the ADF filter is programmed as follows:

- The input signal is coming from a serial interface ($Dsin_{RSFLT}$ = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to - 12 dB.
- RSFLT enabled, and the decimation by four is enabled.
- HPF is enabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a 22-bit max signal to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$Asout_{SCALE} = 24^5 \times 10^{\frac{-12}{20}} \times 1 = 2.10^6$$

$$DS_{SCALE} = \frac{\ln(2.10^6)}{\ln(2)} + 1 = 21.93 \text{ bits}$$

If a higher gain is used, the RSFLT may saturate the output signal for strong input signals.

At the end, the final signal amplitude is:

$$Asout_{HPF} = 24^5 \times 10^{\frac{-12}{20}} \times 10^{\frac{9.5}{20}} \times 1 = 5.9711 \times 10^6$$

$$Dsout_{HPF} = \left(\frac{\ln(5.9711 \times 10^6)}{\ln(2)} + 1\right) = 23.51 \text{ bits}$$

or:

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.51}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

27.7.6 How to compute SAD thresholds

The SAD does not compute the RMS value of the converted signal, but the average of the absolute values. As a consequence, the estimated level differs from the RMS value of the signal:

- For a sine signal having an RMS value of 1, the SAD computes a level of 0.9.
- For a white or pink noise signal having an RMS value of 1, the SAD computes a level of about 0.8.

Note: FRSIZE[2:0] has a big influence on the accuracy of the level estimation: big FRSIZE[2:0] values give better results.

Threshold programming with SADMOD = 01

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 63 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02.

The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 63 dBSPL produces a digital signal of about:
- 26 dBFS - (94 - 63) = - 57 dBFS.

- SCALE value adjustment

  For this example, the filter configuration is the following:
  - CIC5 with a decimation by 16
  - RSFLT enabled with a decimation by 4
  - HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. As DFTL0 provides samples only used for a sound detection (samples not provided to the application), a bigger gain value can be applied: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 15.6 dB).
- Input signal amplitude

  The input signal is - 57 dBFS, corresponding to an amplitude

  Asin = $10^{(-57/20)}$ = 0.00141 LSB.
- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 16^5 \times 10^{\frac{15.6}{20}} \times 10^{\frac{9.5}{20}} \times \frac{1}{256} = 73.68 \times 10^3 \text{ or } 97.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.00141 \times G_{SAD} \sim 104 \text{ LSB}$$

  The gain can be increased if the expected amplitude is too small. For the targeted application, 104 LSB is fine.

  If the input signal is expected to be a sine, the sound level for a signal amplitude of 104 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 66 \text{ LSB}$$

  where 0.9 is the correction factor to apply with respect to the RMS value.
- Program the trigger value

ANMIN and SNTHR must be programmed to trigger the SAD when the input signal level reaches 66 LSB.

For SADMOD[1:0] = 01, the threshold value is given by:

$$THRH = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

where GdB_{SNTHR} represents the decibel value selected by SNTHR[3:0].

When SNTHR[3:0] = 6 dB for example, this formula becomes:

$$THRH = 2 \times ANMIN$$

So ANMIN = THRH/2 = 66 /2 = 33 LSB.

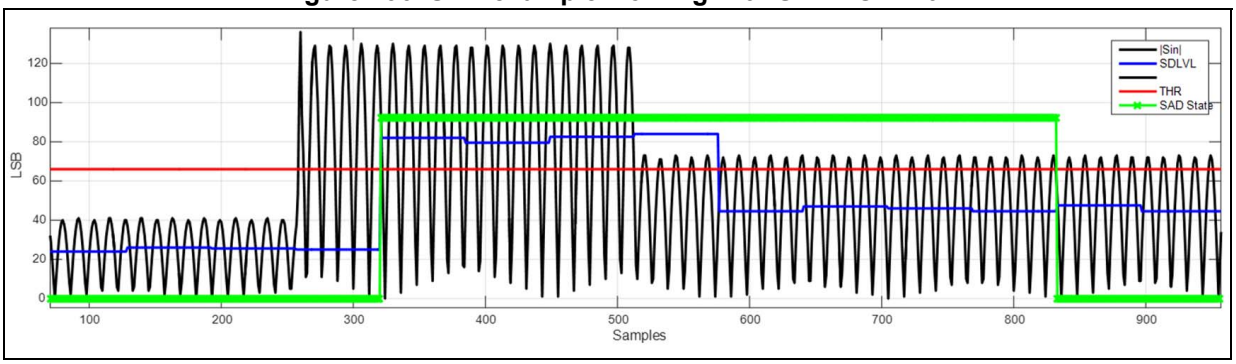
In *Figure 235*, the trigger value (THRH in red) is fixed to 66 LSB. The input signal is at - 65dBFS during 256 samples, then its value goes to - 55 dB for 256 samples, and finally it is reduced to - 60 dBFS.

The blue curve is showing the sound level estimation (SDLVL) versus time. Fluctuation on the estimated value can be observed due to windowing effect of FRSIZE samples.

The SAD DETECT state (when green signal is high) is maintained during four additional frames due to hangover function value.

In this example ANSLP = FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 33.

Figure 236. SAD example working with SADMOD = 01



Threshold programming with SADMOD = 1x

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 57 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02. The sensitivity of this microphone is typically - 26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 57 dBSPL produces a digital signal of about:
- 26 dBFS - (94-63) = - 63 dBFS.

- Adjust SCALE value

  For this example, the filter configuration is the following:

  – CIC4 with a decimation by 48

– RSFLT bypassed
– HPF enabled

A SCALE value of 3.5 dB is recommended for this configuration. The samples provided by DFTL0 are only used for a sound detection, without providing the samples to the application, a bigger gain value can be provided: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 24 dB).

- Input signal amplitude

  The input signal is - 63 dBFS, corresponding to an amplitude:

  Asin = $10^{(-63/20)}$ = 0.000708 LSB.

- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 48^4 \times 10^{\frac{24}{20}} \times 0.003906 = 328.6 \times 10^3 \text{ or } 110.3 \text{ dB}$$

The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.000708 \times G_{SAD} \sim 232 \text{ LSB}$$

The gain can be increased if the expected amplitude is too small.

If the input signal is expected to be a sine, the sound level for a signal amplitude of 232 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 148 \text{ LSB}$$

where 0.9 is the correction factor to apply with respect to the RMS value.

Note: ANLVL converges to average of SDLVL values, with a long constant time.
So SDLVL ~ ANLVL = 148 LSB for a constant input signal at 57 dBSPL.

- Programming trigger value

  For SADMOD = 1', the SAD compares the estimated ambient noise multiplied by the gain selected by SNTHR[3:0] to ANMIN[12:0] * 4.

  For simplification, SNTHR[3:0] is set to 1 (6 dB), meaning that ANLVL is multiplied by two.

  The SAD triggers if 2 * ANLVL > THRH.

  In this mode,

$$THRH = 4 \times ANMIN$$

So the SAD triggers if:

$$ANLVL > 2 \times ANMIN$$

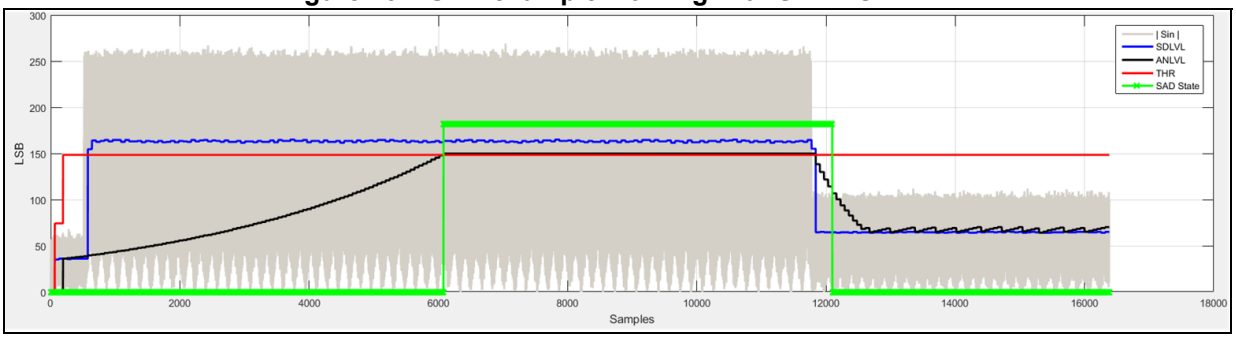
So ANMIN = 148 / 2 = 74 LSB

In *Figure 236*, the trigger value (THRH in red) is fixed to 148 LSB. The input signal is at - 75 dBFS during 512 samples, then its value goes to - 62 dB for 11000 samples, and finally it is reduced to - 70 dBFS.

The blue curve shows the sound level estimation (SDLVL) versus time. The black curve shows the ambient noise estimation versus time, increasing or decreasing logarithmically. During the learning phase, it reaches the SDLVL value.

In this example ANSLP = 6, FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 74.

Figure 237. SAD example working with SADMOD = 1x



27.8 ADF registers

All the ADF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

27.8.1 ADF global control register (ADF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also used to synchronize several ADF together by controlling the adf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on the adf_trgo signal and triggers the acquisition on enabled filter having their ACQMOD[2:0] = 01x and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

27.8.2 ADF clock generator control register (ADF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The clock adf_proc_ck must be enabled before enabling other ADF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. Ii is used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (see *Section 27.4.13: Register protection* for details).

The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two 2 periods of adf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This bitfield is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{adf_itf_ck} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0: adf_ker_ck provided to the SITF

1: adf_ker_ck / 2 provided to the SITF

2: adf_ker_ck / 3 provided to the SITF

...

127: adf_ker_ck / 128 provided to the SITF

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the ADF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the ADF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the ADF_CCK is given by the following formula:

$$F_{ADF_CCK} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0000: The ADF_CCK clock is adf_proc_ck.

0001: The ADF_CCK clock is adf_proc_ck / 2.

0010: The ADF_CCK clock is adf_proc_ck / 3.

...

1111: The ADF_CCK clock is adf_proc_ck / 16.

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter. This bitfield is not significant if the CKGMOD = 0.

000x: TRGO selected

0010: adf_trg1 selected

Others: Reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, the sensitivity is forced to falling edge, thus TRGSENS value is not taken into account. This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: ADF_CCK1 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK1 pin.

0: The ADF_CCK1 pin direction is in input.

1: The ADF_CCK1 pin direction is in output.

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 5 **CCK0DIR**: ADF_CCK0 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK0 pin.

0: The ADF_CCK0 pin direction is in input.

1: The ADF_CCK0 pin direction is in output.

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if the filter is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: ADF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK1 pin.

Bit 1 **CCK0EN**: ADF_CCK0 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK0 pin

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

27.8.3 ADF trigger input selection register (ADF_TRGISEL)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used to select the trigger signal.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGISEL[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRGISEL[2:0]**: Input selection for adf_trgi

This bitfield is used to select the signal connected to adf_trgi input. The filter(s) must be disabled (DFTEN=0) before changing this field.

000: adf_trgi0 is selected (default after reset)

001: adf_trgi1 is selected

010: adf_trgi2 is selected

011: adf_trgi3 is selected

100: adf_trgi4 is selected

Others: Reserved

Refer to *Table 224* to find signals connected to each input.

27.8.4 ADF serial interface control register 0 (ADF_SITF0CR)

Address offset: 0x080

Reset value: 0x0000 1F00

This register is used to control the serial interface SITF0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD[1:0]		Res.	SCKSRC[1:0]		SITFEN
			rw	rw	rw	rw	rw			rw	rw		rw	rw	rw

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when SITFACTIVE is set to 0 (see *Section 27.4.13: Register protection* for details).
The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
0: The serial interface is not active, and can be configured if needed.
1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (see *Manchester mode* for details on computation).
In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This bitfield is set and cleared by software. It is used to define the serial interface type.
00: LF_MASTER SPI mode
01: Normal SPI mode
10: Manchester mode: rising edge = logic 0, falling edge = logic 1
11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This bitfield is set and cleared by software. It is used to select the clock source of the serial interface.
00: Serial clock source is ADF_CCK0.
01: Serial clock source is ADF_CCK1.
others: reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is set and cleared by software. It is used to enable/disable the serial interface.

0: Serial interface disabled

1: Serial interface enabled

27.8.5 ADF bitstream matrix control register 0 (ADF_BSMX0CR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to DFLT0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
											rw	rw	rw	rw	rw

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when BSMXACTIVE is set to 0. This BSMXACTIVE flag cannot go to 0 if DFLT0 is enabled.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream selection

This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLT0.

00000: bs0_r provided to DFLT0

00001: bs0_f provided to DFLT0

others: reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

27.8.6 ADF digital filter control register 0 (ADF_DFLT0CR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.
r	r			rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: DFLT0 active flag

This bit is set and cleared by hardware. It indicates if DFLT0 is active: can be running or waiting for events.

0: DFLT0 not active (can be re-enabled again, via DFLTEN bit, if needed)

1: DFLT0 active

Bit 30 **DFLTRUN**: DFLT0 run status flag

This bit is set and cleared by hardware. It indicates if DFLT0 is running or not.

0: DFLT0 not running and ready to accept a new trigger event

1: DFLT0running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This bitfield is set and cleared by software. It is used to define the number of samples to be discarded every time DFLT0 is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **TRGSRC[3:0]**: DFLT0 trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger DFLT0.

0000: TRGO selected

0010: adf_trgi selected

Others: Reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: DFLT0 trigger sensitivity selection

This bitfield is set and cleared by software. It is used to select the trigger sensitivity of the external signals

0: A rising edge event triggers the acquisition.

1: A falling edge even triggers the acquisition.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge.
This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: DFLT0 trigger mode

This bitfield is set and cleared by software. It is used to select the filter trigger mode.

000: Asynchronous continuous acquisition mode

001: Asynchronous single-shot acquisition mode

010: Synchronous continuous acquisition mode

011: Synchronous single-shot acquisition mode

100: Window continuous acquisition mode

Others: Same as 000

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold.

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 0 **DFLTEN**: DFLT0 enable

This bit is set and cleared by software. It is used to control the start of acquisition of the DFLT0 path. This bit behavior depends on ACQMOD[2:0] and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x.

27.8.7 ADF digital filer configuration register 0 (ADF_DFLT0CICR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to control the main CIC filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD8
						rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCICD[7:0]								Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (see *Table 229* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

000000: 0 dB
000001: + 3.5 dB
000010: + 6 dB or shift left by 1 bit
...
011000: + 72 dB or shift left by 12 bits
100000: - 48.2 dB or shift right by 8 bits (default value)
100001: - 44.6 dB
100010: - 42.1 dB or shift right by 7 bits
100011: - 38.6 dB
...
101110: -6 dB or shift right by 1 bit
101111: -2.5 dB
Others: Reserved

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection

This bitfield is set and cleared by software.It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
511: Decimation ratio is 512.

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC order

This bitfield is set and cleared by software. It is used to select the order of the MCIC.

100: MCIC configured in single $Sinc^4$ filter
101: MCIC configured in single $Sinc^5$ filter
Others: Reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter

This bitfield is set and cleared by software.

0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

27.8.8 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
						rw	rw	rw			rw				rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency

This bitfield is set and cleared by software. it is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.

00: Cut-off frequency = 0.000625 x FPCM

01: Cut-off frequency = 0.00125 x FPCM

10: Cut-off frequency = 0.00250 x FPCM

11: Cut-off frequency = 0.00950 x FPCM

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 **HPFBYP**: High-pass filter bypass

This bit is set and cleared by software. It is used to bypass the high-pass filter.

0: HPF not bypassed (default value)

1: HPF bypassed

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio

This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter

0: Decimation ratio is 4 (default value).

1: Decimation ratio is 1.

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass

This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.

0: Reshape filter not bypassed (default value)

1: Reshape filter bypassed

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

27.8.9 ADF delay control register 0 (ADF_DLY0CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **SKPBF**: Skip busy flag
This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.
0: ADF ready to accept a new value into SKPDLY[6:0]
1: Last valid SKPDLY[6:0] still under precessing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream
This bitfield is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.
0: No input sample skipped
1: 1 input sample skipped
...
127: 127 input samples skipped

27.8.10 ADF DFLT0 interrupt enable register (ADF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
		rw	rw	rw	rw	rw								rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLIE**: SAD sound-level value ready enable
This bit is set and cleared by software.
0: Sound-level-ready interrupt disabled
1: Sound-level-ready interrupt enabled

Bit 12 **SDDETIE**: Sound activity detection interrupt enable
This bit is set and cleared by software.
0: Sound-trigger interrupt disabled
1: Sound-trigger interrupt enabled

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

27.8.11 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for the digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1						r		rc_w1	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLF**: Sound level value ready flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that new sound level value is not ready. Write 0 has no effect.
1: Read 1 means that new sound level value is ready. Write 1 clears this flag.

Bit 12 **SDDETF**: Sound activity detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no sound activity is detected. Write 0 has no effect.
1: Read 1 means that sound activity is detected. Write 1 clears this flag.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RXNEF**: RXFIFO not empty flag
This bit is set and cleared by hardware according to the RXFIFO level.
0: RXFIFO empty
1: RXFIFO not empty

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DOVRF**: Data overflow flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no overflow is detected. Write 0 has no effect.
1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag
This bit is set by hardware, and cleared by the hardware when the RXFIFO level is lower than the threshold.
0: RXFIFO threshold not reached
1: RXFIFO threshold reached

27.8.12 ADF SAD control register (ADF_SADCR)

Address offset: 0x0B8

Reset value: 0x0000 0000

This register is used for the configuration and the control of the sound activity detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
		rw	rw		rw	rw	rw	rw		r	r	rw	rw	rw	rw

Bit 31 **SADACTIVE**: SAD Active flag

This bit is set and cleared by hardware. It is used to check if the SAD is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the SADACTIVE is set to 0 (see *Section 27.4.13: Register protection* for details).

The delay between a transition on SADEN and a transition on SADACTIVE is two periods of AHB clock and two periods of adf_proc_ck.

0: SAD not active and can be configured if needed

1: SAD active and protected fields cannot be configured.

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:12 **SADMOD[1:0]**: SAD working mode

This bitfield is set and cleared by software. It is used to define the way the SAD works.

00: Threshold value computed according to the estimated ambient noise

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a voice activity detector.

01: Threshold value equal to ANMIN[12:0], multiplied by the gain selected by SNTHR[3:0]

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a sound detector.

1x: Threshold value given by 4 x ANMIN[12:0]

The SAD triggers when the estimated ambient noise (ANLVL), multiplied by the gain selected by SNTHR[3:0] is bigger than the defined threshold. In this mode, the SAD is working like an ambient noise estimator. Hysteresis function cannot be used in this mode.

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **FRSIZE[2:0]**: Frame size

This bitfield is set and cleared by software. it is used to define the size of one frame and also to define how many samples are taken into account to compute the short-term signal level.

000: 8 PCM samples used to compute the short-term signal level

001: 16 PCM samples used to compute the short-term signal level

010: 32 PCM samples used to compute the short-term signal level

011: 64 PCM samples used to compute the short-term signal level

100: 128 PCM samples used to compute the short-term signal level

101: 256 PCM samples used to compute the short-term signal level

11x: 512 PCM samples used to compute the short-term signal level

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 **HYSTEN**: Hysteresis enable

This bit is set and cleared by software. It is used to enable/disable the hysteresis function (see *Table 229* for details). This bit must be kept to 0 when SADMOD[1:0] = 1x.

0: Hysteresis function disabled. THR_H is always used.

1: Hysteresis function enabled. THR_H is used for MONITOR to DETECT transition and THR_L is used for DETECT to MONITOR transition.

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **SADST[1:0]**: SAD state

This bitfield is set and cleared by hardware. It indicates the SAD state and is meaningful only when SADEN = 1. The SAD state can be:
- LEARN when the SAD is in learning phase or in SDLVL computation mode
- MONITOR when the SAD is in monitoring phase
- DETECT when the SAD detects a sound

00: SAD in LEARN state
01: SAD in MONITOR state
11: SAD in DETECT state

Bit 3 **DETCFG**: Sound trigger event configuration

This bit is set and cleared by software. It is used to define if the sddet_evt event is generated only when the SAD enters to MONITOR state or when the SAD enters or exits the DETECT state.
0: sddet_evt generated when SAD enters the MONITOR state
1: sddet_evt generated when SAD enters or exits the DETECT state

Note: This bit can be write-protected (see Section 27.4.13: Register protection for details).

Bits 2:1 **DATCAP[1:0]**: Data capture mode

This bitfield is set and cleared by software. It is used to define in which conditions, the samples provided by DLFT0 are stored into the memory.
00: Samples from DFLT0 not transfered into the memory
01: Samples from DFLT0 transfered into the memory when SAD is in DETECT state
1x: Samples from DFLT0 transfered into memory when SAD and DFLT0 are enabled

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 0 **SADEN**: Sound activity detector enable

This bit is set and cleared by software. It is used to enable/disable the SAD.
0: SAD disabled and SAD state reset
1: SAD enabled

27.8.13 ADF SAD configuration register (ADF_SADCFGR)

Address offset: 0x0BC

Reset value: 0x0000 0000

This register is used for the configuration of the sound activity detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ANMIN[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HGOVR[2:0]			Res.	LFRNB[2:0]			Res.	ANSLP[2:0]			SNTHR[3:0]			
	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **ANMIN[12:0]**: Minimum noise level

This bitfield is set and cleared by software. It is used to define the minimum noise level and then the sensitivity. It represents a positive number.

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HGOVR[2:0]**: Hangover time window

This bitfield is set and cleared by software. Once the SAD state is DETECT, this parameter is used to define the amount of time the sound is allowed to remain below the threshold, before switching the SAD to MONITOR state (see FRSIZE bitfield for the description of a frame).

000: SAD back to MONITOR state if sound is below threshold for 4 frames
001: SAD back to MONITOR state if sound is below threshold for 8 frames
010: SAD back to MONITOR state if sound is below threshold for 16 frames
011: SAD back to MONITOR state if sound is below threshold for 32 frames
100: SAD back to MONITOR state if sound is below threshold for 64 frames
101: SAD back to MONITOR state if sound is below threshold for 128 frames
110: SAD back to MONITOR state if sound is below threshold for 256 frames
111: SAD back to MONITOR state if sound is below threshold for 512 frames

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LFRNB[2:0]**: Number of learning frames

This bitfield is set and cleared by software. It is used to define the number of learning frames to perform the first estimate of the noise level.

000: 2 frames used to compute the initial noise level
001: 4 frames used to compute the initial noise level
010: 8 frames used to compute the initial noise level
011: 16 frames used to compute the initial noise level
1xx: 32 frames used to compute the initial noise level

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ANSLP[2:0]**: Ambient noise slope control

This bitfield is set and cleared by software. It is used to define the positive and negative slope of the noise estimator, in charge of updating the ANLVL (see *Ambient noise estimation (ANLVL)* for information about programming this bitfield).

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

Bits 3:0 **SNTHR[3:0]**: Signal to noise threshold

This bitfield is set and cleared by software. It is used to define THR_H (and THR_L if hysteresis is enabled). See *Table 229* for details.

0000: THR_H is 3.5 dB higher than ANLVL
0001: THR_H is 6.0 dB higher than ANLVL
0010: THR_H is 9.5 dB higher than ANLVL
0011: THR_H is 12 dB higher than ANLVL
0100: THR_H is 15.6 dB higher than ANLVL
0101: THR_H is 18 dB higher than ANLVL
0110: THR_H is 21.6 dB higher than ANLVL
0111: THR_H is 24.1 dB higher than ANLVL
1000: THR_H is 27.6 dB higher than ANLVL
1001: THR_H is 30.1dB higher than ANLVL
others: Reserved

Note: This bitfield can be write-protected (see Section 27.4.13: Register protection for details).

27.8.14 ADF SAD sound level register (ADF_SADSDLVR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register contains the short-term sound-level computed by the SAD.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SDLVL[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **SDLVL[14:0]**: Short term sound level

This bitfield is set by hardware. It contains the latest sound level computed by the SAD. To refresh this value, SDLVLF must be cleared.

27.8.15 ADF SAD ambient noise level register (ADF_SADANLVR)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register contains the ambient noise level computed by the SAD.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ANLVL[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **ANLVL[14:0]**: Ambient noise level estimation

This bitfield is set by hardware. It contains the latest ambient noise level computed by the SAD. To refresh this bitfield, the SDLVLF flag must be cleared.

27.8.16 ADF digital filter data register 0 (ADF_DFLT0DR)

Address offset: 0x0F0

Reset value: 0x0000 0000

This register is used to read the data processed by the digital filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:8 **DR[23:0]**: Data processed by DFT0

Bits 7:0 Reserved, must be kept at reset value.

27.8.17 ADF register map

Table 242. ADF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADF_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO
	Reset value																																0
0x004	ADF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV [3:0]				TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0
0x008 - 0x07C	Reserved	Reserved																															
0x020	ADF_TRGISELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGISEL[2:0]		
	Reset value																														0	0	0
0x080	ADF_SITF0CR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]				Res.	Res.	SITFMOD [1:0]		Res.	SCKSRC [1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084	ADF_BSMX0CR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0

Table 242. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADF_DFLT0CR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.	TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD [2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0					0	0	0	0				0		0	0	0		0	0	0
0x08C	ADF_DFLT0CICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090	ADF_ DFLT0RSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 - 0x0A0	Reserved	Reserved																															
0x0A4	ADF_DLY0CR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8	Reserved	Reserved																															
0x0AC	ADF_DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
	Reset value																			0	0	0	0	0	0	0			0		0	0	0
0x0B0	ADF_DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
	Reset value																			0	0	0	0	0						0		0	0
0x0B4	Reserved	Reserved																															
0x0B8	ADF_SADCR	SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
	Reset value	0																		0	0		0	0	0	0		0	0	0	0	0	0
0x0BC	ADF_SADCFGR	Res.	Res.	Res.	ANMIN[12:0]													Res.	HGOVR [2:0]			Res.	LFRNB [2:0]			Res.	ANSLP[2:0]			Res.	SNTHR [2:0]		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0C0	ADF_SADSDLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADF_SADANLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8 - 0x0EC	Reserved	Reserved																															

Table 242. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F0	ADF_DFLT0DR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Refer to *Section 2.3* for the register boundary addresses.

28 Touch sensing controller (TSC)

28.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

28.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 21 capacitive sensing channels without active shield
- Up to 7 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

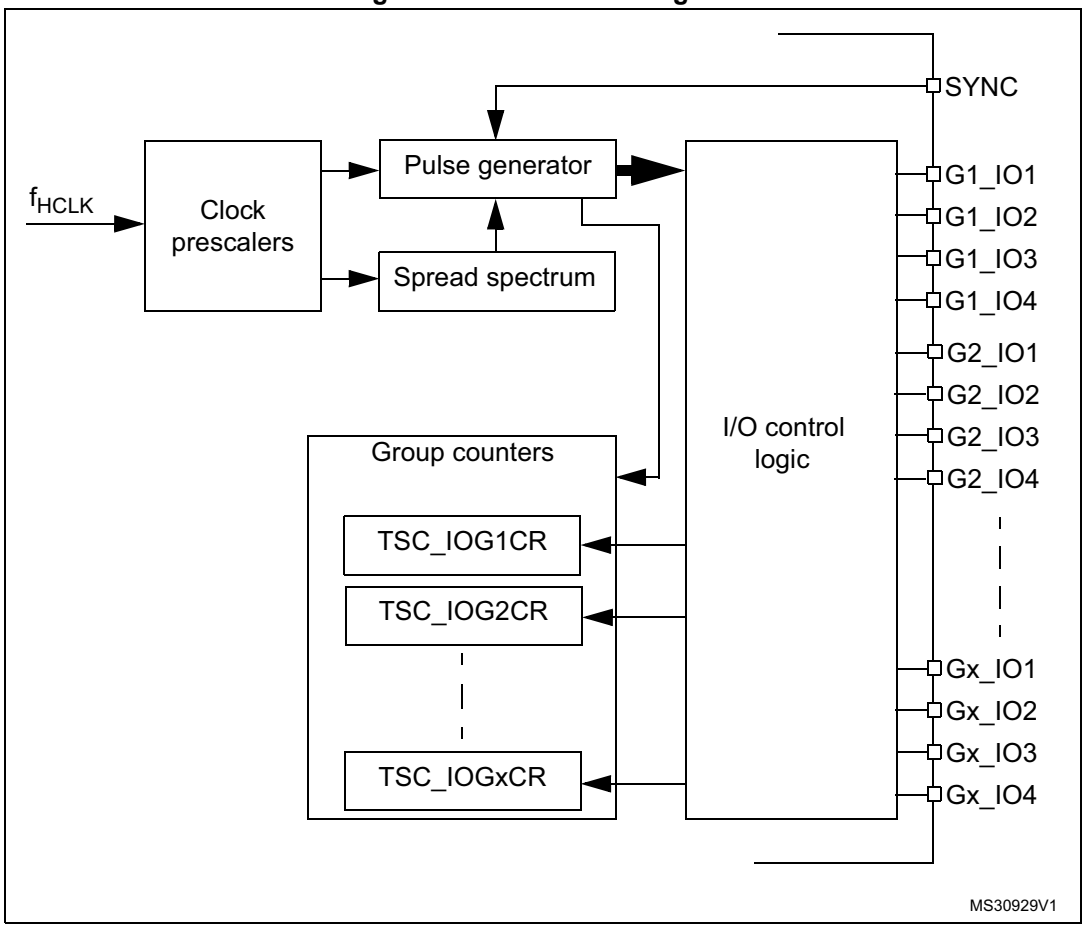
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to I/O availability.

28.3 TSC functional description

28.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in *Figure 238*.

Figure 238. TSC block diagram



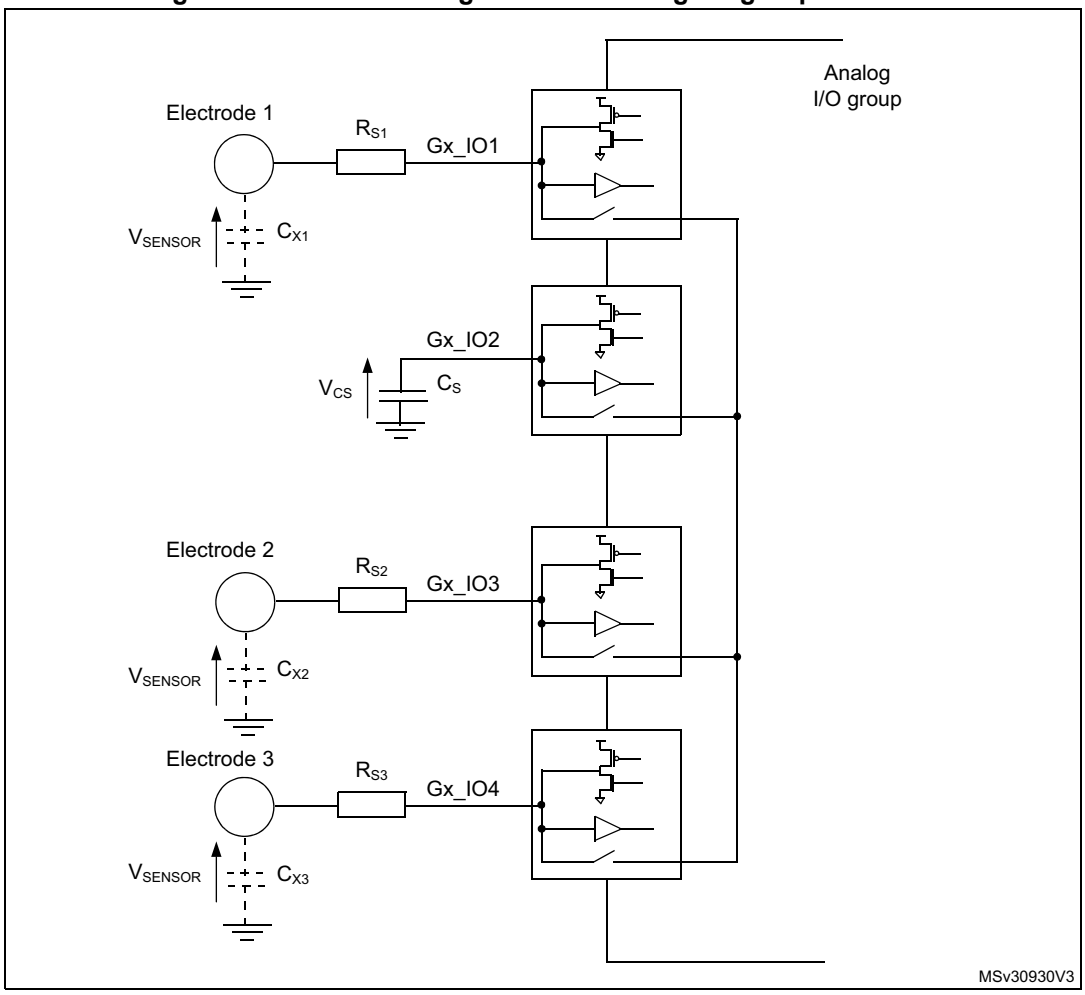
28.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group composed of up to four GPIOs (see *Figure 239*). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S. Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

Figure 239. Surface charge transfer analog I/O group structure



Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

Table 243 details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.

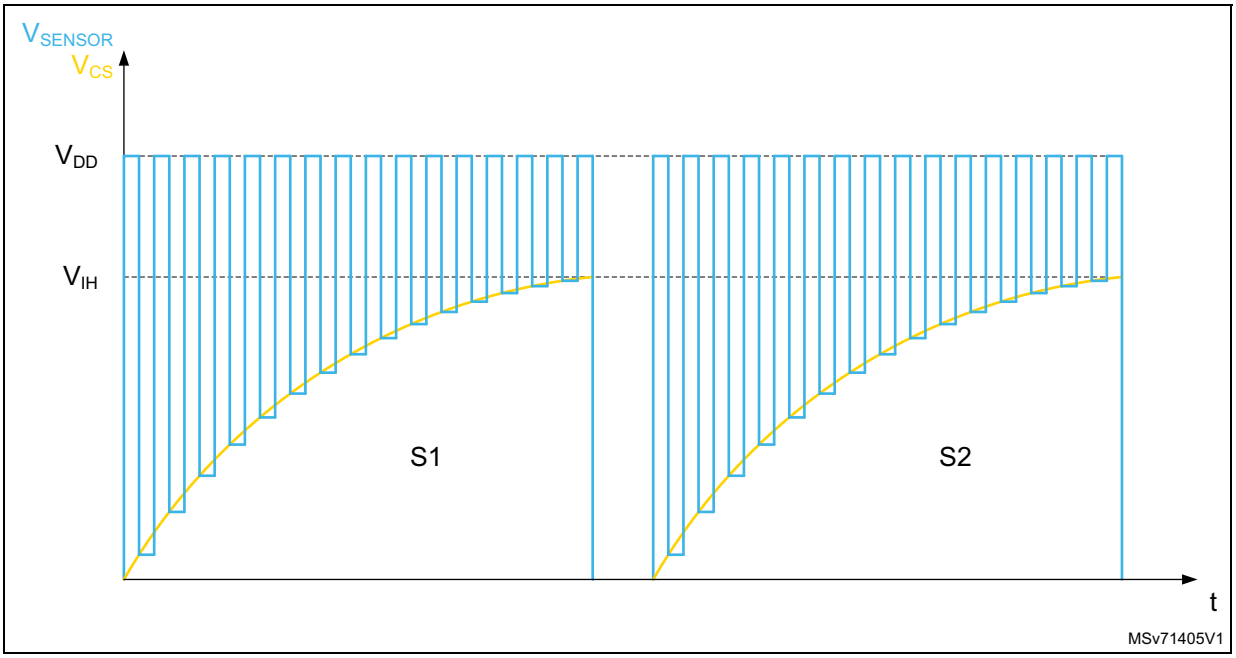
Table 243. Acquisition sequence summary

State	Gx_IO1 (channel)	Gx_IO2 (sampling)	Gx_IO3 (channel)	Gx_IO4 (channel)	State description
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all C_X and C_S
#2	Input floating				Dead time
#3	Output push-pull high	Input floating			Charge C_{X1}
#4	Input floating				Dead time
#5	Input floating with analog switch closed		Input floating		Charge transfer from C_{X1} to C_S
#6	Input floating				Dead time
#7	Input floating				Measure C_S voltage

Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The voltage variation over the time on the sampling capacitor C_S is detailed below (refer to *Figure 239* for V_{SENSOR} and V_{CS} definition):

Figure 240. Sampling capacitor voltage variation



28.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

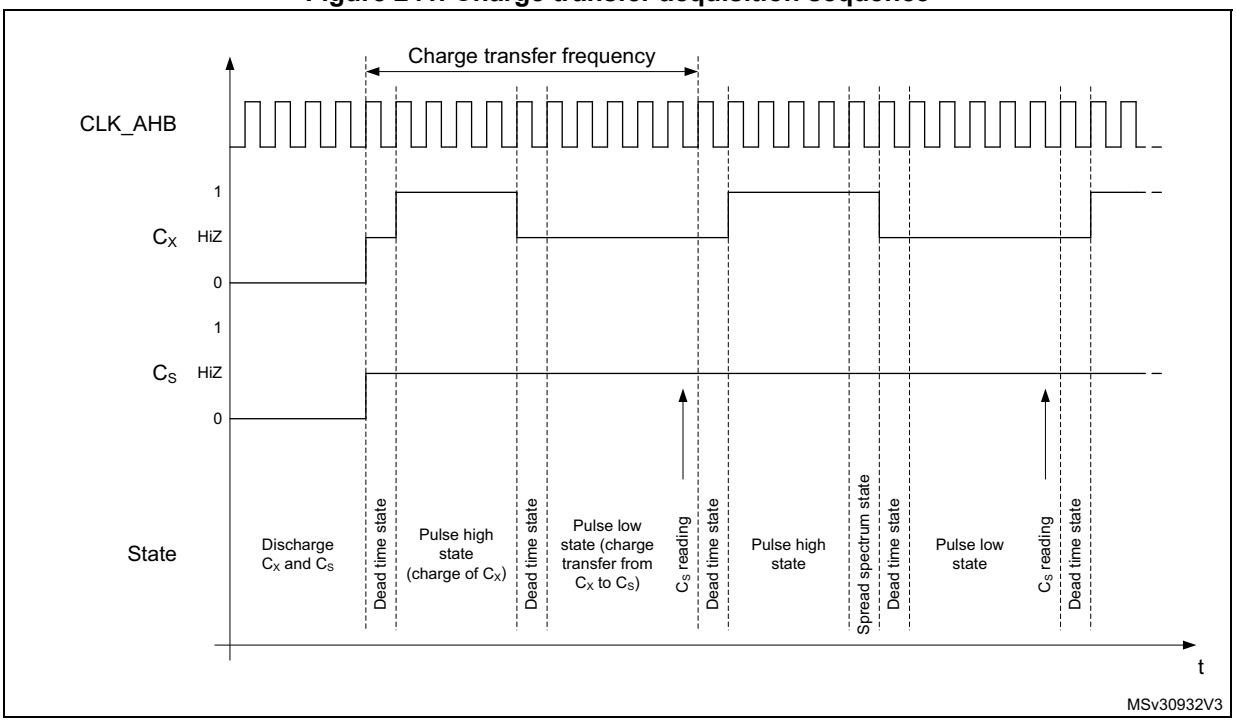
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

The reset and clock controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to *Section 10: Reset and clock control (RCC)*.

28.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in *Figure 241*.

Figure 241. Charge transfer acquisition sequence



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_X) and the pulse low state (transfer of charge from C_X to C_S) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 µs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_X is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 1 period of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

The reading of the sampling capacitor I/O, to determine if the voltage across C_S has reached the given threshold, is performed at the end of the pulse low state.

Note: The following TSC control register configurations are forbidden:

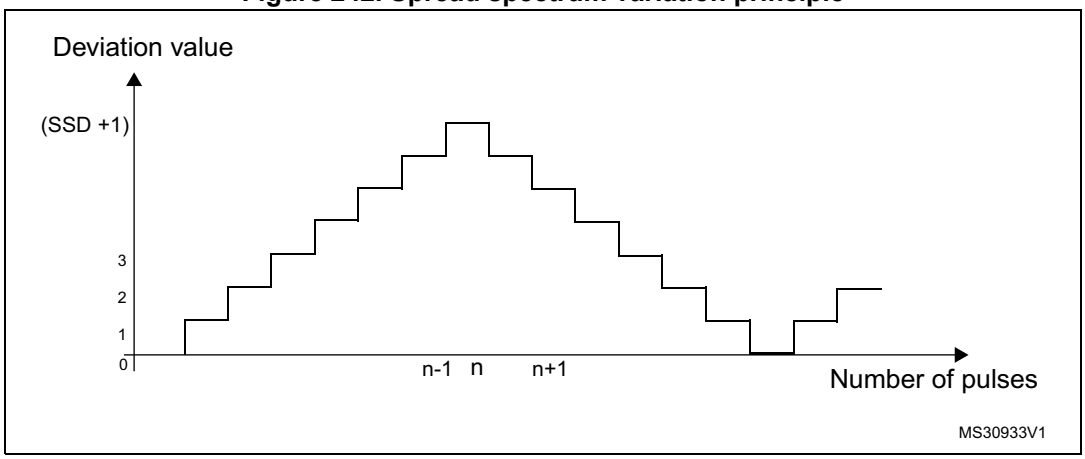
- *bits PGPSC are set to 000 and bits CTPL are set to 0001*
- *bits PGPSC are set to 001 and bits CTPL are set to 0000*

28.3.5 Spread spectrum feature

The spread spectrum feature generates a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 µs), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 242. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 244. Spread spectrum deviation versus AHB clock frequency

f_{HCLK}	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

28.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

28.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

Table 245. I/O state depending on its mode and IODEF bit value

IODEF bit	Acquisition status	Unused I/O mode	Channel I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	Ongoing	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	Ongoing	Input floating	-	-

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IOy bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

Note: During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

28.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAIC and MCEIC bits in the TSC_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

28.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller is able to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_IOy bit in the TSC_IOHCR register.

28.4 TSC low-power modes

Table 246. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.

28.5 TSC interrupts

Table 247. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	Yes	No	No
Max count error	MCEIE	MCEIF	MCEIC	Yes	No	No

28.6 TSC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

28.6.1 TSC control register (TSC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of C_X).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from C_X to C_S).

0000: 1x t_{PGCLK}
0001: 2x t_{PGCLK}
...
1111: 16x t_{PGCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 28.3.4: Charge transfer acquisition sequence for details.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000: 1x t_{SSCLK}
0000001: 2x t_{SSCLK}
...
1111111: 128x t_{SSCLK}

Note: These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled
1: Spread spectrum enabled

Note: This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0: f_{HCLK}
1: f_{HCLK} /2

Note: This bit must not be modified when an acquisition is ongoing.

Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler

These bits are set and cleared by software.They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000: f_{HCLK}
001: f_{HCLK} /2
010: f_{HCLK} /4
011: f_{HCLK} /8
100: f_{HCLK} /16
101: f_{HCLK} /32
110: f_{HCLK} /64
111: f_{HCLK} /128

Note: These bits must not be modified when an acquisition is ongoing.

Note: Some configurations are forbidden. Refer to the Section 28.3.4: Charge transfer acquisition sequence for details.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255
001: 511
010: 1023
011: 2047
100: 4095
101: 8191
110: 16383
111: reserved

Note: These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low
1: I/Os are in input floating

Note: This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only
1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

Note: This bit must not be modified when an acquisition is ongoing.

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the ongoing acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSCE**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

Note: When the touch sensing controller is disabled, TSC registers settings have no effect.

28.6.2 TSC interrupt enable register (TSC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled

1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled

1: End of acquisition interrupt enabled

28.6.3 TSC interrupt clear register (TSC_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a 0 has no effect.

0: No effect

1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a 0 has no effect.

0: No effect

1: Clears the corresponding EOAF of the TSC_ISR register

28.6.4 TSC interrupt status register (TSC_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAIC of the TSC_ICR register.

0: Acquisition is ongoing or not started

1: Acquisition is complete

28.6.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

28.6.6 TSC I/O analog switch control register (TSC_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.

0: Gx_IOy analog switch disabled (opened)

1: Gx_IOy analog switch enabled (closed)

Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).

28.6.7 TSC I/O sampling control register (TSC_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_IOy sampling mode

These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx_IOy unused

1: Gx_IOy used as sampling capacitor

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

28.6.8 TSC I/O channel control register (TSC_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_IOy channel mode

These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.

0: Gx_IOy unused

1: Gx_IOy used as channel

Note: These bits must not be modified when an acquisition is ongoing.

During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

28.6.9 TSC I/O group control status register (TSC_IOGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7S	G6S	G5S	G4S	G3S	G2S	G1S
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7E	G6E	G5E	G4E	G3E	G2E	G1E
									rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started

1: Acquisition on analog I/O group x is complete

Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

28.6.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (x = 1 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CNT[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across C_S has reached the threshold).

28.6.11 TSC register map

Table 248. TSC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	TSC_CR	CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE	SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV [2:0]			IODEF	SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x0004	TSC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
	Reset value																															0	0
0x0008	TSC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
	Reset value																															0	0
0x000C	TSC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
	Reset value																															0	0
0x0010	TSC_IOHCR	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0014	Reserved																																
0x0018	TSC_IOASCR	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	Reserved																																
0x0020	TSC_IOSCR	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	Reserved																																
0x0028	TSC_IOCCR	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																
0x0030	TSC_IOGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7E	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x0034	TSC_IOG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	TSC_IOG2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 248. TSC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x003C	TSC_IOG3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	TSC_IOG4CR	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	TSC_IOG5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	TSC_IOG6CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	TSC_IOG7CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 Coupling and chaining bridge (CCB)

29.1 CCB introduction

The coupling and chaining bridge (CCB) is a sophisticated security mechanism designed to ensure that private keys are protected, even against CPU access, and are wrapped with unique device keys, and stored securely in a secure storage.

The CCB can be programmed to implement special coupling and chaining operations required to protect private keys used in PKA protected operations.

These coupling and chaining operations involve the PKA, the SAES, and sometimes the RNG peripherals.

29.2 CCB main features

- AHB system slave port, mapping multiple peripherals (see *Section 29.3*)
- AHB configuration slave port (CCB peripheral)
  - All read accesses supported
  - 32-bit word write accesses only (otherwise an AHB error occurs)
- Optional coupling of PKA RAM writes to SAES input data register
- Support for multiple optional chaining options:
  - Read and write chaining between PKA RAM and SAES
  - SAES to CCB read chaining, with comparison to a 128-bit reference tag
  - RNG output chaining with either PKA RAM or SAES_IVR registers
- Dedicated sequences to support three PKA protected operations (modular exponentiation, scalar multiplication, ECDSA signature)
  - One-time sequence to prepare a PKA protected operation (blob creation)
  - Many-time sequence to execute PKA protected operation (blob usage)
- Optional private key generation for ECDSA signature, and for ECC private key cryptography (key never accessible to the application in clear-text)
- Possibility to encrypt, with AES-256, any PKA blob encryption key using the device unique secret key DHUK (in the SAES), which makes the PKA encrypted blob usable only on this device
- Software-reset capability
- Repository for cryptographic subsystem tamper event flags

29.3 CCB implementation

The CCB connects the peripherals listed in *Table 249*.

Table 249. Peripherals connected by CCB interconnect

Peripheral	Coupling and chaining feature(1)	Tamper status flag in CCB_SR
RNG	Yes	TAMP_EVT0
SAES	Yes	TAMP_EVT1
PKA	Yes	TAMP_EVT2 (registers + PKA RAM)
		TAMP_EVT3 (PKA operations)
CCB	Yes	TAMP_EVT4
AES	No	TAMP_EVT5
HASH	No	-

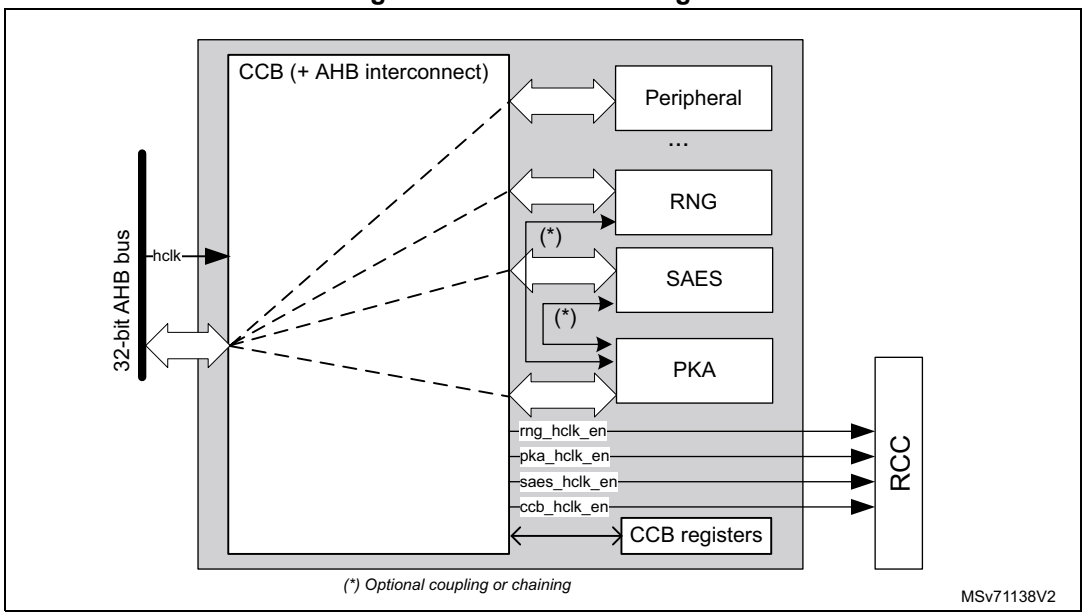
1. See *Section 29.4.4* for details.

29.4 CCB functional description

29.4.1 CCB block diagram

Figure 243 shows the CCB block diagram. Peripherals connected to the CCB interconnect, with optional coupling/chaining features, are defined in *Section 29.3*.

Figure 243. CCB block diagram



29.4.2 CCB internal signals

Table 250 describes the relevant internal signals interfacing the CCB peripheral. Signals targeting peripherals connected to the CCB are not listed.

Table 250. CCB internal input/output signals

Signal name	Signal type	Description
hclk	Digital input	Interconnect system clock input
ccb_hclk_en	Digital output	CCB clock request to the RCC
x_hclk_en	Digital output	Peripheral x clock request to the RCC (x = SAES, PKA, or RNG)

29.4.3 CCB reset and clocks

The CCB shares the same clock used by the AHB bus. Once the CCB clock is activated in the RCC, and coupling and chaining activity is activated by setting CCOP ≠ 0 in CCB_CR, the CCB automatically requests the RCC to enable RNG, SAES, and PKA clocks. These clocks remain enabled until the application gates them in the RCC, with CCOP cleared in the CCB_CR register.

The CCB has a dedicated reset bit in the RCC, and a system reset input. Locally to the CCB it is also possible to trigger a reset of the peripheral based on IPRST (do not use it when the application does not use CCB) in the CCB_CR register (refer to the bitfield description for details). Status register CCB_SR has a specific reset behavior (see *Section 29.7.2*).

When at least a cryptographic peripheral raises a tamper event, the CCB requests its clock to the RCC. The tamper status TAMP_EVTi of these peripherals can be found in the CCB_SR register, as defined in *Section 29.3*.

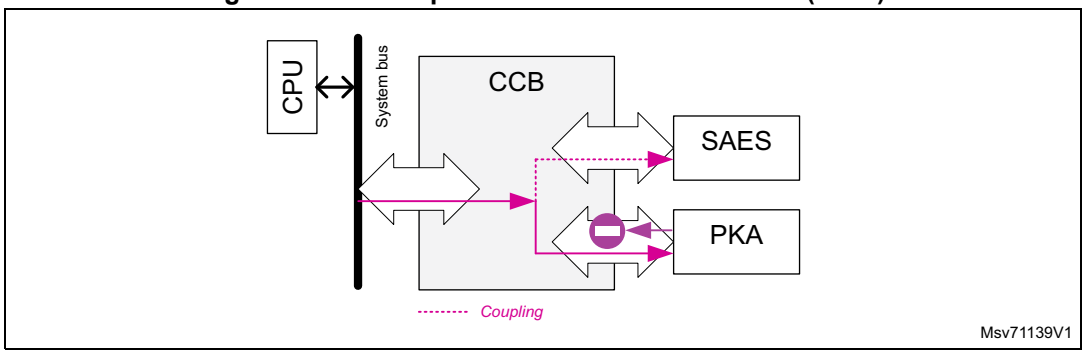
29.4.4 CCB coupling and chaining

Note: The selection of these chaining modes is defined in Section 29.5. In most of these operations, a read of the PKA RAM returns 0 (shown with a purple circle in the following figures).

Coupling CPU → PKA + SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and sends them to SAES_DINR, as shown in *Figure 244*. Write to SAES_DINR is not coupled to the PKA. The CCB sends both data and address to SAES_DINR.

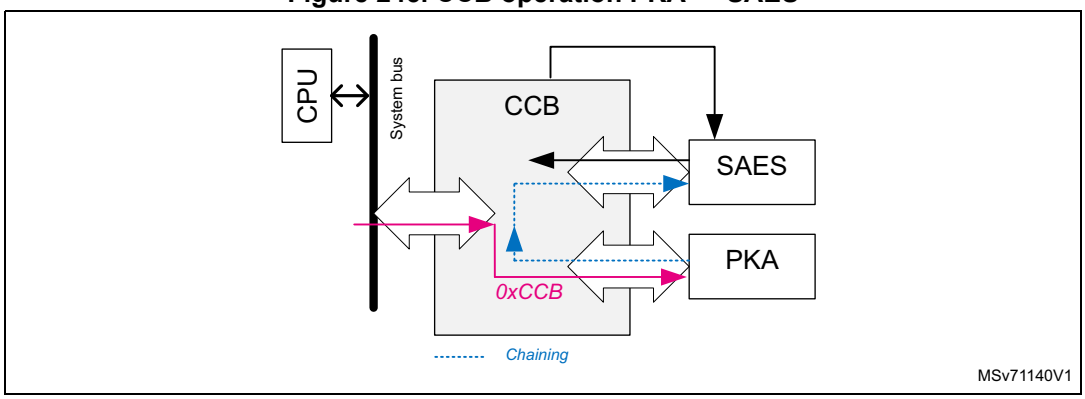
Figure 244. CCB operation CPU → PKA + SAES (write)



Chaining PKA → SAES

When the corresponding OPSTEP is selected, for each 32-bit write request to the PKA RAM of magic number 0xCCB, the CCB generates a read request from the write target (in the PKA RAM) to SAES_DINR.

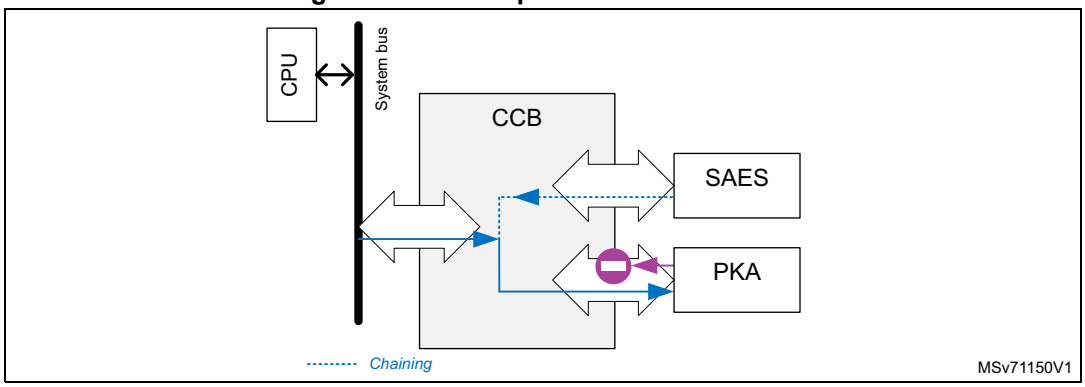
Figure 245. CCB operation PKA → SAES



Chaining SAES → PKA

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data by a 32-bit word coming from SAES_DOUTR. When available, this chaining is activated when the application writes 0xCCB to PKA RAM.

Figure 246. CCB operation SAES → PKA

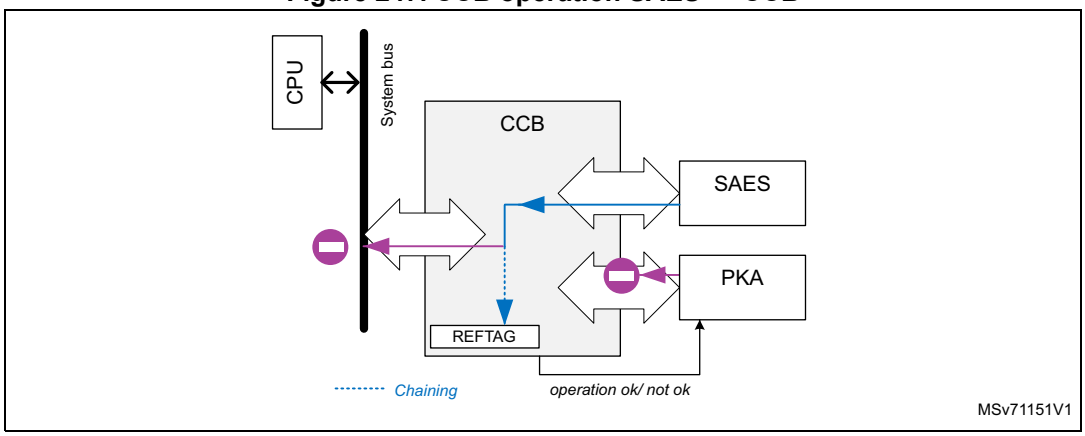


Chaining SAES → CCB

When the corresponding OPSTEP is selected, the CCB takes the read requests to SAES_DOUTR, and sends them to the system interconnect and to CCB_REFTAGRx holding the reference TAG information. For confidentiality reason, read data toward system bus are zeroed by the CCB.

If the CCB finds that the transfered value matches its reference information, it signals to the PKA that it can proceed with the protected PKA operation (START bit can be set).

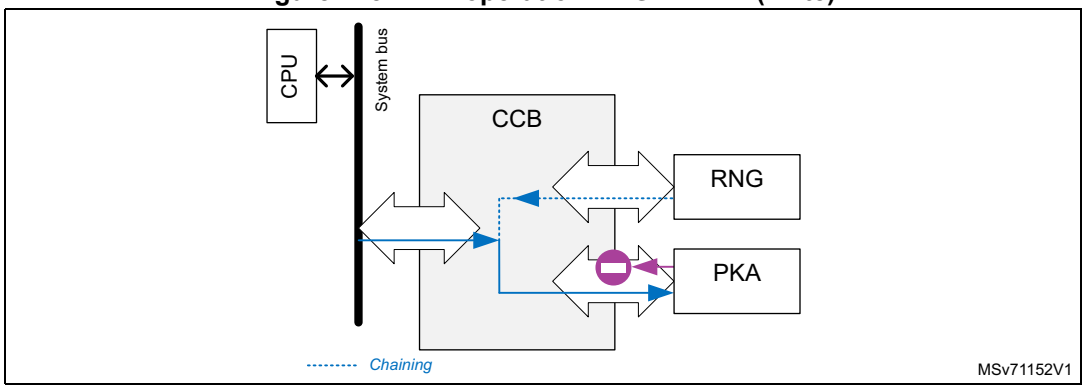
Figure 247. CCB operation SAES → CCB



Chaining RNG → PKA (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to the PKA RAM, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

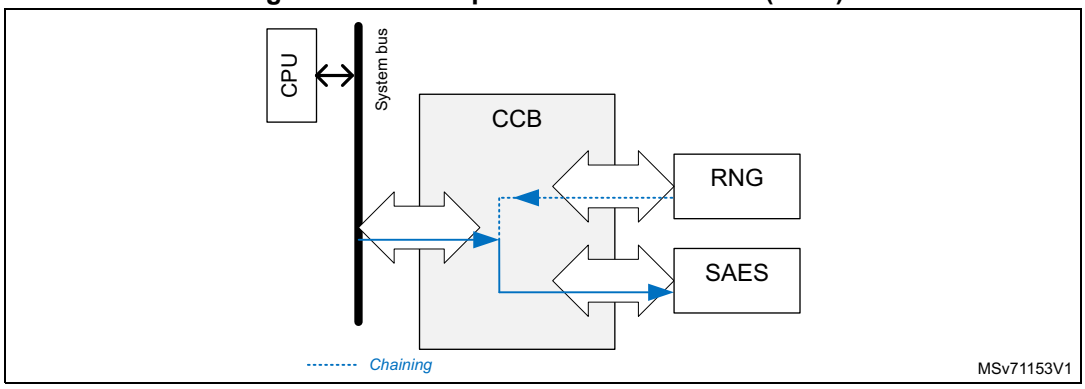
Figure 248. CCB operation RNG → PKA (write)



Chaining RNG → SAES (write)

When the corresponding OPSTEP is selected, the CCB takes the write requests that go to SAES_IVR1 to IVR3, and substitutes these data with random numbers coming from RNG_DR. For consistency, the application can write 0xCCB to activate this chaining.

Figure 249. CCB operation RNG → SAES (write)

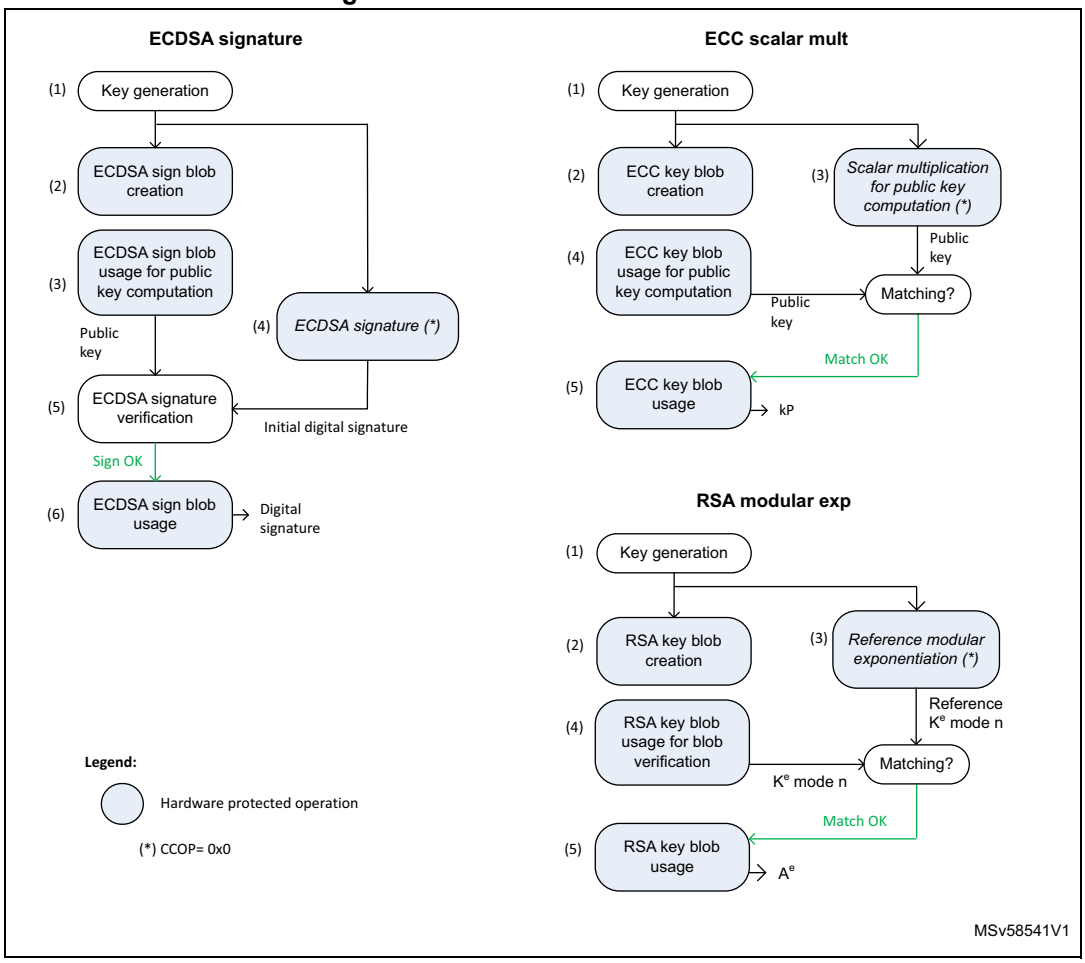


29.5 CCB coupling and chaining operations

When an application must enforce private key confidentiality by wrapping them with a hardware unique key, the following steps are needed:

1. Select the protected operation in the PKA that uses this private key (ECDSA signature, scalar multiplication, modular exponentiation).
2. When CCB is active (OPSTEP=0x1), select in the SAES the DHUK (with optional XOR with BHK), or unwrap in the SAES an existing blob encryption key that has been encrypted (in wrapped-key mode) using the DHUK. A 256-bit KEYSIZE must be selected.
   Note that the DHUK value is different when CCB is active or inactive.
3. When the application uses this protected operation in the PKA, it must load the related blob information in the PKA RAM and in SAES_DINR.
   - If the information exactly matches the conditions of the blob generation (choice and order of the writes of protected parameters in the PKA RAM, selection of blob encryption key, use of original encrypted secrets), the CCB allows the application to set START bit in PKA_CR.
   - If the blob information does not match the conditions of the blob generation, the application can only reset the sequence by setting IPRST bit in CCB_CR. CCB, PKA, and SAES become available when CCB_BUSY is cleared in CCB_SR.
4. At any time, the application can abort the executing protected operation by using the IPRST bit in CCB_CR.When creating a blob, always use it once before trusting it to be stored in non-volatile memory or given to a less trusted code (see *Figure 250*). More specifically, each time a blob is created:
   - For ECDSA signature blob: use it to compute the public key, and use this key to verify the signature computed during the blob creation.
   - For ECC key blob: use it to compute the public key, and verify key coordinates match those of the key computed during the blob creation.
   - For RSA key blob: use it to compute a reference modular exponentiation and verify it matches the one computed during the blob creation.

Figure 250. Blob creation scenarios



This section describes how to do the above steps, for all available protected operations in the PKA. Note that:

- **Bold** elements in the following tables are key inputs that the application must save in non-volatile memory after the newly created blob has been confirmed by its first usage.
- The CCB automatically goes to the next step, updating the OPSTEP value, when the application clears or performs the end-step event described in tables of *Section 29.5.2* to *Section 29.5.4*.
- When an incremental read or write is mentioned, the application must write four bytes in the PKA memory, with an incremental memory address (modulo 0x8). Otherwise, the CMF error flag is triggered, and the process must be restarted. See *Section 34: Public key accelerator (PKA)* for details.
- When writing to SAES_CR register the application must write the configuration expected by the CCB, otherwise an OPERR might be raised. Refer to the OPSTEP "CPU action" instructions in the application tables (*Table 251* through *Table 257*).
- When the CCB uses the RNG, the application must check in RNG_SR that no error occurred (SEIS = CEIS = 0), and that a random number is ready (DRDY = 1). In case of issue, RNGOKF is not set, and the application must ensure that the RNG is functional before running again the OPSTEP sequence that uses the RNG.

- For each OPSTEP>0, the application must manage the error flags that could occur in the RNG, SAES or PKA blocks. During a normal protected operation, no error flags are expected.
- When CCB draws random number to compute ECC private keys, it implements the testing candidates method defined by NIST in FIPS PUB 186-4 section B.4.

Note: It is not possible to generate RSA key pairs with the CCB, they must be generated outside the device.

29.5.1 Preliminary steps before any protected operation

The following steps are required before running any operation described in this section:

1. Initialize the GTZC using the same security/privilege configuration for the SAES, PKA, RNG and CCB peripherals (see *Section 5: Global TrustZone controller (GTZC)*). Otherwise OPERR=0x1A occurs (see *Section 29.7.2: CCB status register (CCB_SR)*).
2. Activate both RNG clocks, then initialize the peripheral to be fully functional, without pending errors.
3. Activate the PKA clock, then release the PKA reset. The PKA is functional once its RAM is automatically erased, and it has fetched the necessary random numbers. For further details refer to enabling/disabling PKA in *Section 34: Public key accelerator (PKA)*.
4. Activate the SAES clock, then release the SAES reset. The SAES is functional once it has fetched necessary random numbers.
5. Set IPRST bit in CCB_CR, then clear it when CCB_BUSY=0x0 in CCB_SR.

Note: Step 4 can be done before step 3, as CCOP = 0x0.

29.5.2 ECDSA signature

This PKA operation is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0xC3, the private key d can be manipulated by the application only if encrypted, using a special blob called “ECDSA signature blob”.

ECDSA signature blob creation

The ECDSA signing operation is prepared by generating and then encrypting the private key d, and computing the reference GCM tag. This process is summarized *Table 251*, with coupling and chaining operations described in *Section 29.4.4*. This sequence starts by

generating the ECDSA private key before setting CCOP to 0xC0 in CCB_CR register (usage of 0xC2 value is not recommended).

Table 251. Blob creation steps for ECDSA signature (CCOP = 0b110000<i>0)

OP STEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in the step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set AES-GCM up in 256-bit encryption mode, with the DHUK or the decrypted key in register (if applicable).
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to SAES_IVR3 are loaded with randoms. – Read back **blob IV** in SAES_IVR registers.
0x3	0x24	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x6 (<i>=1)	0x24	0x2	RNG -> PKA	RNGOKF set	Write incrementally d in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if sanity check fails).
0x4 (<i>=0)			n/a	DATAOKF set	Write incrementally d in the PKA.
0x8	0x24		PKA -> SAES	DATAOKF set, GCMPH = 0x3	– Read incrementally d from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. Special rule must be used for the last 64-bit word (see step 7 after the table). – Read **encrypted d** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0xA		0x3	N/A	IPRST set in the CCB	– Complete GCM final. Read **reference tag** in the SAES.In this step keep the SAES enabled (EN bit set).
0x0	N/A	N/A	N/A	N/A	– Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. – Verify the newly created blob, running the *ECDSA signature blob use for public-key computation* with it.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.

The sequence described in *Table 251* is detailed below. Refer to the table for OPSTEP values and related coupling/chaining modes.

1. Write CCOP = 0xC0 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1 (repeat if the EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK

(KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1 or 0x2). See *Section 31: AES hardware accelerator (AES)* for details.

4. In the SAES, set the GCM chaining mode in encryption mode (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x0, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write SAES_IVRx, with any value (except SAES_IVR0 that must be equal to 0x2). For SAES_IV1 to IV3, random values are loaded into the SAES by the CCB. If an error occurs, RNGERRF is set in CCB_SR.
   c) Read back SAES_IVR1 to SAES_IVR3, and **save the value** in non-volatile memory after blob verification.
   d) Set EN in SAES_CR, and wait until CCF = 1 in SAES_ISR (end of GCM init).
   e) Clear CCF flag in SAES_ICR.
5. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, as described in *Section 34: Public key accelerator (PKA)*. Do the following when writing data in the PKA:
   a) Write two 32-bit word in the PKA RAM, then wait until CCF is set in SAES_ISR. An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF in SAES_ICR.
6. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the private key *d* in the PKA RAM using the following sequence:
   a) If CCOP = 0xC0, write the private key d value in the PKA RAM. If CCOP = 0xC2, write the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB then draws the value of *d* from the RNG, with PKA running a special sanity check to verify random compliance with NIST standard.
   b) Complete parameter information with the mandatory two 32-bit zeroed words. At the end of this process, the private key d of size *d_size* is written in the PKA RAM. Clear the error flag, then restart step 6 if the PKA sets RNGERRF when BUSY is cleared.
7. When the PKA sets its completion flag (RNGOKF if CCOP = 0xC2, DATAOKF if CCOP = 0xC0), clear it, then read the private key d in the PKA RAM using the sequence below (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) For each 64-bit words in the PKA RAM, write twice the 32-bit magic number 0xCCB. The target content of the PKA memory is transfered to SAES_DINR by the CCB. Wait until CCF is set in SAES_ISR (end of AES encryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear CCF flag in SAES_ICR. Then read the encrypted private key d in SAES_DOUT, **to save it** in the non-volatile memory after blob verification.
   c) For the final 64-bit word (mandatory zeroed word): if the *d_size* is proportional to 16 bytes (for example, for 192- and 521-bit curves), the final 64 bits must be written to with two 32-bit magic numbers 0xCCB. If the *d_size* is not proportional to 16 bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
8. When the DATAOKF completion flag is set (private key d OK), and BUSY is cleared in SAES_SR, select the final phase of the running GCM chaining mode (GCMPH = 0x3), after having cleared DATAOKF. Write the last GCM block in SAES_DINR, then wait until CCF is set in SAES_ISR (end of GCM final).

9. Read SAES_DOUTR four times to retrieve the ECDSA signature blob reference tag. Save it in the non-volatile memory after blob verification. Do not disable SAES by clearing the EN bit.
10. Clear CCOP by setting IPRST bit in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP is written different from 0.

ECDSA signature blob usage for ECDSA signature

Table 252 summarizes the ECDSA signing operation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 251*. Coupling and chaining traffic options are described in *Section 29.4.4*. This sequence starts by setting CCOP to 0xC3 in the CCB_CR register.

Table 252. Blob use steps for ECC signature (CCOP = 0b11000011)

OPSTEP	MODE in PKA	GCMPH	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, by writing in PKA_CR MODE=0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt the blob encryption key in SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write hash z.
0x12		0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13	0x24	0x1	CPU → PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES → PKA	DATAOKF set	– Write encrypted d in SAES_DINR, and runs AES decryption round. – Write incrementally private key d in the PKA, using 0xCCB magic number. A special rule must be used for the last 64-bit write (see step 7 in the sequence detailed after this table). – Repeat for each 16-byte block of data.
0x16			RNG → PKA	RNGOKF set, GCMPH = 0x3	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x17		0x3	SAES → CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18	0x24	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A		N/A	N/A	N/A	Read **signature (r,s)** in the PKA RAM.

1. The application must clear and/or perform the described event for the CCB to automatically go to the next step.

2. SAES cannot be used for any operations, for security reasons.

The detailed sequence described in *Table 252* is detailed below. Refer to the table for the OPSTEP values and related coupling and chaining modes.

1. Write CCOP = 0xC3 in CCB_CR.
2. Enable the PKA in the correct mode, writing in PKA_CR MODE=0x24 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR.
3. Write the hash of the message in the PKA RAM.
4. Initialize the ECC signature blob key in the SAES. This key can be the DHUK (KEYSEL = 0x1 or 0x4), or a key decrypted in 256-bit wrapped key mode by the DHUK (KEYSEL = 0x1 or 0x4, KMOD = 0x1, MODE = 0x1). Refer to *Section 31: AES hardware accelerator (AES)* for details.
5. Set the GCM chaining mode in decryption in the SAES (CHMOD = 0x3, GCMPH = 0x0, MODE = 0x2, DATATYPE = 0x0). Then:
   a) Configure the 256-bit key size using KEYSIZE.
   b) Write to SAES_IVRx registers the value saved in non-volatile memory.
   c) Set EN in SAES_CR, and wait until CCF is set in SAES_ISR (end of GCM init).
   d) Clear CCF flag in SAES_ICR.
   e) Write to CCB_REFTAGR registers, the ECDSA signature blob reference tag value saved in non-volatile memory.
6. Select the header phase of the running GCM chaining mode (GCMPH = 0x1). Write the protected operation (ECDSA signature) mandatory parameters in the PKA RAM, exactly as done during the blob creation. See step 5 in *ECDSA signature blob creation* for details.
7. Select the payload phase of the running GCM chaining mode (GCMPH = 0x2). Write the protected private key d in the PKA RAM, following the below sequence (starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8):
   a) Write two 64-bit words of encrypted private key d in SAES_DINR. Wait until CCF is set in SAES_ISR (end of AES decryption). An SAES interrupt can be triggered when this flag is set.
   b) Clear the CCF flag in the SAES_ICR register. Then, for both 64-bit words, write four times the 32-bit magic number 0xCCB to the private key d parameter addresses in the PKA memory. The CCB transfers the clear-text content of SAES_DOUTR to the PKA.
   c) For the final 64-bit word (mandatory zeroed word): If *d_size* defined in step 6 in *ECDSA signature blob creation* is not proportional to 16-bytes (for example, for 256- and 512-bit curves), the final 64 bits must be written with 0s.
8. When DATAOKF = 1 in PKA_SR (private key d ok), clear this flag. Then write to the integer k parameter addresses in the PKA memory, starting from the LSB word in the PKA RAM, and increasing the addresses modulo 0x8. The CCB draws the value of k from the RNG. The PKA then runs a special sanity check to verify random compliance

with NIST standard. Clear the error flag, then restart step 7 if the PKA sets RNGERRF when BUSY is cleared.

9. When RNGOKF = 1 in PKA_SR (random k ok), clear this flag. Then select the final phase of the running GCM chaining mode (GCMPH = 0x3). Write the last GCM block in SAES_DINR. Then wait until CCF is set in SAES_ISR (end of GCM final).
10. Read four times SAES_DOUTR (returns 0s). If PKA parameters were written exactly like during the blob creation, and the correct private key d was used, and all expected DATAOKF/RNGOKF flags were raised then cleared, the application can set START bit in PKA_CR to run the ECDSA signature operation. Otherwise, IPRST in CCB_CR must be used to restart the sequence from step 1.
11. Clear CCOP by setting IPRST in CCB_CR. After CCB_BUSY is cleared in CCB_SR, clear the IPRST bit. The SAES, and PKA can again be used for any tasks. They are unprotected by the CCB until CCOP in CCB_CR is written different from 0.

ECDSA signature blob use for public-key computation

The ECC public key computation using the encrypted private key d, the reference blob IV, and the computed GCM tag of *Table 251* is summarized in *Table 253*. The additional ECDSA verification at the end of the sequence must be used before saving a new ECDSA signature blob in non-volatile memory.Shown coupling and chaining traffic options are described in *Section 29.4.4*. This sequence starts by setting CCOP to 0x81 in the CCB_CR register.

Table 253. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable).
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC signature parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 2x32-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted d in SAES DINR and run AES decryption round. – Write incrementally scalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	SAES -> CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18 (2)	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)		N/A	N/A	PROCENDF set	The PKA computes the operation. PKA RAM is read as 0.

Table 253. Blob use steps for signing public key computation (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1A(2)	0x20	N/A	N/A	IPRST set in the CCB	– Read public key point Q in PKA RAM – Set IPRST in CCB
0x0	0x26	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. – Before writing new ECDSA signature blob in non-volatile memory do an initial digital signature, as summarized in *Figure 250.* – Enable the PKA – Write MODE= 0x24 in the PKA – Write hash z & random nonce k in PKA RAM – Write in PKA RAM the same curve parameters and private key d as the current ECDSA signature blob – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If error code= 0xD60D save the computed signature in temporary RAM. – Finally, verify the above generated ECDSA signature. – Write MODE= 0x26 in the PKA – Write the same hash z as above in PKA RAM – Write ECDSA verification parameters in PKA, including the public key coordinates computed in step 0x1A. – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If the signature is valid save the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons

29.5.3 ECC scalar multiplication

This operation consists in the computation of k x P, where P is a point on a curve over prime fields, and “x” is the elliptic curve scalar point multiplication. The computation result is a point that belongs to the same curve, or a point at infinity.

The scalar multiplication operation is protected against side-channel attacks. When CCOP = 0x0 in CCB_CR, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x81, the scalar multiplier k can be manipulated by the application only if encrypted, using a special blob called “ECC key blob”.

Scalar multiplication “ECC key” blob creation

The scalar multiplication operation is prepared by generating then encrypting the scalar multiplier k, and computing the reference GCM tag. This process is summarized in *Table 254*. Coupling and chaining operations described in *Section 29.4.4*. This sequence starts by generating the scalar multiplier k before setting CCOP to 0x80 in CCB_CR register (usage of 0x82 value is not recommended).

Table 254. Blob creation steps for scalar multiplication (CCOP = 0b100000<i>0)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event[1]	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, by writing in PKA_CR MODE=0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable).
0x2		0x0	RNG → SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. Registers IVR1 to 3 are loaded with randoms. – Read back **blob IV** in IVR registers.
0x3	0x20	0x1	CPU → PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in SAES, triggered after each 64-bit write.
0x6 (<i>=1)	0x20	0x2	RNG → PKA	RNGOKF set	Write incrementally k in the PKA, using 0xCCB magic number. Values are drawn from the RNG. The PKA runs a special sanity check to verify random compliance with NIST standard (RNGERRF is set if the sanity check fails).
0x4 (<i>=0)			N/A	DATAOKF set	Write incrementally k in the PKA.
0x8	0x20	0x2	PKA → SAES	DATAOKF set, GCMPH = 0x3	– Read incrementally k from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted k** in SAES DOUTR. – Repeat for each 16-byte block of data.
0xA		0x3	N/A	IPRST set in the CCB	– Complete GCM final. Read **reference tag** in the SAES.In this step keep the SAES enabled (EN bit set).
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. Before writing new ECC key blob in non-volatile memory do a public key computation, as summarized in *Figure 250*. – Enable the PKA – Write MODE= 0x20 in the PKA – Write base point G coordinates as input in PKA RAM – Write in PKA RAM the same curve parameters and scalar multiplier k as the current ECC key blob – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If error code= 0xD60D save this reference public key in temporary RAM. Finally, verify the newly created blob by using it for public key computation (see *Table 255*)

1. The application must clear and/or perform described event for CCB to automatically go to the next step.

Scalar multiplication blob use for example for ECC decryption

ECC scalar multiplication computation using the encrypted k, the reference blob IV and the computed GCM tag of *Table 254* is summarized in *Table 255*. Shown coupling and chaining traffic options are described in *Section 29.4.4*. This sequence starts by setting CCOP to 0x81 in CCB_CR register.

Table 255. Blob use steps for scalar multiplication (CCOP = 0b10000001)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, by writing in PKA_CR MODE=0x20 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally, decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write input point P coordinates in the PKA RAM. When computing a public key, write basepoint G coordinates.
0x12	0x20	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CBB_REFTAGR.
0x13		0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write ECC scalar multiplication parameters in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES -> PKA	DATAOKF set, GCMPH = 0x3	– Write encrypted k in SAES_DINR, and runs AES decryption round. – Write incrementally scalar multiplier k in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	N/A	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18(2)	0x20	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19 (2)				PROCENDF set	The PKA computes the operation. The PKA RAM is read as 0.
0x1A(2)				N/A	Read **kP point coordinates** in the PKA RAM. If this point is the public key verify it matches the reference public key computed in *Table 254* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

29.5.4 RSA modular exponentiation

The modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod n.

This modular exponentiation operation (MODE = 0x3) is protected against side-channel attacks. When CCOP = 0x0, all parameters loaded in the PKA RAM are managed in the clear by the application. When CCOP = 0x45, the exponent e and phi value can only be manipulated encrypted by application, using a special blob called “RSA key blob”.

Modular exponentiation “RSA key” blob creation

Modular exponentiation operation is prepared by generating then encrypting the scalar multiplier *k* and computing the reference GCM tag. This process is summarized in *Table 256*. The coupling and chaining operations described in *Section 29.4.4*. This sequence starts by setting CCOP to 0x44 in CCB_CR register.

Table 256. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, writing in PKA_CR MODE= 0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in 256-bit encryption mode with DHUK or the decrypted key in register (if applicable).
0x2		0x0	RNG -> SAES	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in encryption mode. SAES_IVR1 to 3 registers are loaded with randoms. – Read back blob IV in SAES_IVR registers.
0x3	0x03	0x1	CPU -> PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES triggered after each 2x32-bit write.
0x4	0x3	0x2	N/A	DATAOKF set	Write incrementally clear-text exponent e.
0x5				DATAOKF set	Write incrementally clear-text phi.
0x8			PKA -> SAES	DATAOKF set	– Read incrementally e from the PKA (writing 0xCCB magic numbers), and runs AES encryption rounds. – Read **encrypted e** in SAES_DOUTR. – Repeat for each 16 bytes block of data.
0x9				DATAOKF set, GCMPH = 0x3	– Read incrementally phi from the PKA (writing 0xCCB magic numbers), and run AES encryption rounds. – Read **encrypted phi** in SAES_DOUTR. – Repeat for each 16 bytes block of data

Table 256. Blob creation steps for modular exponentiation (CCOP = 0b01000100)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action in step
0xA	0x3	0x3	N/A	IPRST set in the CCB	– Complete GCM final. – Read **reference tag** in the SAES. In this step keep the SAES enabled (EN bit set).
0x0	N/A	N/A	N/A	N/A	Clear the IPRST bit in CCB when CCB_BUSY and OPSTEP are cleared in CCB_SR. Before writing new ECC key blob in non-volatile memory do an public key computation, as summarized on *Figure 250*. – Enable the PKA – Write MODE= 0x44 in the PKA – Write a constant K as operand A in the PKA RAM – Write in PKA RAM the same base parameters, Kphi and exponent e as the current RSA key blob – Start the operation setting START bit in PKA_CR – Wait for PROCENDF bit is set. If error code= 0xD60D save this reference Ke in temporary RAM. Finally, verify the newly created blob by using it for Ke computation (see *Table 257*).

1. The application must clear and/or perform described event for CCB to automatically go to the next step.

Modular exponentiation blob use (for example for RSA decryption)

The modular exponentiation computation using the encrypted *e* and *phi*, the reference blob IV, and the computed GCM tag of *Table 256* is summarized on *Table 257*. Shown coupling and chaining traffic options are described in *Section 29.4.4*. This sequence starts by setting CCOP to 0x45 in CCB_CR register.

Table 257. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x1	N/A	N/A	N/A	AES GCM selected, GCMPH = 0x0	– Enable the PKA in the correct mode, by writing in PKA_CR MODE=0x3 and EN=1 (repeat if EN bit stays clear). Wait for INITOK=1 in PKA_SR. – Optionally decrypt blob encryption key in the SAES, using DHUK in 256-bit wrapped key mode. – Set up AES-GCM in decryption mode with DHUK or the decrypted key in register (if applicable). – Write operand A in the PKA RAM.Write constant K as operand A defined in *Table 256* when verifying the created blob.

Table 257. Blob use steps for modular exponentiation (CCOP = 0b01000101)

OPSTEP	MODE in PKA	GCMPH in SAES	Coupling or chaining operations	End-step event(1)	CPU action
0x12	0x03	0x0	N/A	GCMPH = 0x1 (following GCM initialization)	– Complete GCM initialization in decryption mode, using saved random IV. – Write reference tag in CCB_REFTAGR.
0x13		0x1	CPU → PKA + SAES	GCMPH = 0x2	Write protected modular exponentiation parameters (PKA mode 0x3) in the PKA RAM, clearing CCF flags in the SAES, triggered after each 64-bit write.
0x14		0x2	SAES → PKA	DATAOKF set	– Write encrypted e in SAES_DINR, and runs AES decryption round. – Write incrementally exponent e in the PKA, using 0xCCB magic number. – Repeat for each 16-byte block of data.
0x15				DATAOKF set, GCMPH = 0x3	– Write encrypted phi in SAES_DINR, and runs AES decryption round. – Write incrementally phi in the PKA, using 0xCCB magic number. – Repeat for each 16 bytes block of data.
0x17		0x3	SAES → CCB	Tag matches REFTAGR?	– Complete GCM final. – Read computed tag in the SAES (returns 0).
0x18(2)	0x03	N/A	N/A	START = 0x1	Start the operation in PKA setting START bit in PKA_CR.
0x19(2)				PROCENDF set	The PKA computes the operation, PKA RAM is read as 0.
0x1A				N/A	Read $\mathbf{A^e}$ **mod** n in the PKA RAM. If the result is K^e mod n verify it matches the reference computed in *Table 256* before saving the created blob in non-volatile memory.

1. The application must clear and/or perform described event for CCB to automatically go to the next step.
2. SAES cannot be used for any operations, for security reasons.

29.5.5 CCB error

When the CCB detects an error during a coupling and chaining operation (see *Section 29.5*), it sets accordingly OPERR[5:0] in CCB_SR (see *Section 29.7.2* for the possible errors).

When OPERR[5:0] ≠ 0 in CCB_SR, the CCB raises IPRST in CCB_CR, sets OPSTEP[4:0] to 0x1F, and sets CCB_BUSY to 1. The application must clear IPRST to be able to use again the CCB and the peripheral(s) used by the coupling and chaining operation.

29.6 CCB interrupts

The CCB does not support any interrupt.

29.7 CCB registers

The following section provides information on registers.

Reserved bits (marked as Res.) must be kept at their reset values, unless otherwise specified.

For general information relative to register description, refer to Document conventions.

These registers are accessible through the AHB interface of the CCB (32-bit words only). All reads are supported. Writes accesses must be 32-bit word only (other access types generate an AHB error).

29.7.1 CCB control register (CCB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Following a CCB hardware reset, the above reset value applies if OPERR[5:0] = 0x0 and no TAMP_EVT bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: CCB reset

Set this bit to start or abort any PKA protected operation defined in *Section 29.5*. IPRST bit must be low while writing to any CCB register.

After setting this bit, the application must clear it when CCB_BUSY and OPSTEP are cleared in CCB_SR (IPRST clear is ignored otherwise). After this clear all CCB registers are read with their reset values.

Setting IPRST clears any data stored in the SAES and the PKA.

When IPRST and CCB_BUSY are set, SAES registers are not accessible (read as 0, write ignored), and writes to EN and START bits in PKA_CR are ignored.

This bit is set by CCB when OPERR[5:0] ≠ 0 or any TAMP_EVT bit is set.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **CCOP[7:0]**: Coupling and chaining operation

These bits must be set according to the chosen PKA protected operation, see *Section 29.5* for details.

0x044: Modular exponentiation blob creation with priv key from the CPU

0x045: Modular exponentiation blob use

0x080: ECC scalar multiplication blob creation with priv key from the CPU

0x082: ECC scalar multiplication blob creation with priv key from the RNG

0x081: ECC scalar multiplication blob use, or ECDSA signature blob use for public key computation

0x0C0: ECDSA signature blob creation with priv key from the CPU

0x0C2: ECDSA signature blob creation with priv key from the RNG

0x0C3: ECDSA signature blob use for ECDSA signature

Note: When an operation not listed here is written by the application, the write to CCOP[7:0] is ignored. All these operations implement protections against side-channel attacks.

Write to CCOP is ignored when CCB_BUSY is set in CCB_SR.

29.7.2 CCB status register (CCB_SR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is reset by a system reset, or after clearing IPRST in CCB_CR. The CCB reset input from the RCC has no effect on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY
		r	r	r	r	r	r								r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
		r	r	r	r	r	r				r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **TAMP_EVTi**: Tamper i flag (i = 5 to 0)

0: No event

1: Tamper event i has been raised. Global tamper event is raised. See *Section 29.3* for details on the events.

This bit is cleared by writing 0 to IPRST bit in CCR_CR. When at least one TAMP_EVT bit is set, CCB sets IPRST bit.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **CCB_BUSY**: CCB busy

This flag is set when CCB is busy, or when OPSTEP[4:0] ≠ 0 following a CCB reset (hardware reset or use of IPRST in CCB_CR).

0: CCB idle

1: CCB busy or PKA RAM is being cleared following a peripheral reset.

When IPRST and CCB_BUSY are set, the SAES registers are not accessible (write ignored, read returns 0), and writes to EN or START bits in PKA_CR are ignored.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **OPERR[5:0]**: Operation error

When an error is detected during a protected PKA operation, the CCB raises IPRST in CCB_CR, and OPSTEP[4:0] is set to 0x1F. The bitfield is cleared by setting IPRST to 0.

0x0: No error detected during protected PKA operation (if applicable)

0x1: CCB RCC reset has been detected when OPSTEP > 0x0

0x2: SAES RCC reset has been detected when OPSTEP > 0x0

0x3: PKA RCC reset has been detected when OPSTEP > 0x0

0x4: RNG RCC reset has been detected when OPSTEP > 0x0

0x5: The RCC has gated the CCB clock when OPSTEP > 0x0

0x6: The RCC has gated the SAES clock when OPSTEP > 0x0

0x7: The RCC has gated the PKA clock when OPSTEP > 0x0

0x8: The RCC has gated the RNG clock when OPSTEP > 0x0

0x9: Random in SAES_IV1 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xA: Random in SAES_IV2 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xB: Random in SAES_IV3 is trivial (all 0s or all 1s) when OPSTEP = 0x2

0xC: When OPSTEP = 0x17 SAES GCM TAG comparison with CCB_REFTAG failed, or a register different from SAES_DOUT was read in the SAES.

0xD: KEYSIZE ≠ 256 bits in SAES_CR when the SAES is enabled, and OPSTEP > 0x0.

0xE: CHMOD ≠ GCM in SAES_CR when 0x1 < OPSTEP < 0x18, or the SAES is not enabled, or KEYVALID is cleared when GCMPH > 0x0.

0xF: KMOD different from wrapped key mode in SAES_CR when the SAES is enabled, and OPSTEP = 0x1, or 0x1A.

0x10: KEYSEL ≠ 0x1 or 0x4 in SAES_CR when the SAES is enabled, KEYVALID is set, and OPSTEP = 0x1.

0x11: GCMPH not correct (not incremented) in SAES_CR when OPSTEP > 0x1.

0x12: MODE in SAES_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x1.

0x13: BUSY = 1 in SAES_SR when OPSTEP= 0x18 or 0x19. SAES must not be used when PKA is computing a protected operation.

0x14: EN = 0 in PKA_CR when OPSTEP > 0x2 (except 0x12). In other words, PKA must be enabled in any OPSTEP except 0x1, 0x2, or 0x12.

0x15: INITOK = 0 in PKA_SR when OPSTEP > 0x2 (except 0x12).

0x16: Authorized by local firewall, but unexpected AHB error detected when OPSTEP > 0x0.

0x17: LMF = 1 in PKA_SR (only ECDSA verification available) when OPSTEP > 0x0.

0x18: CCB_REFTAGRx registers are equal to 0 when OPSTEP > 0x12.

0x1A: Security or/and privilege protection inconsistent (see *Section 5: Global TrustZone controller (GTZC)*) for the peripherals SAES, PKA, RNG, and CCB when OPSTEP > 0x0. This error is also triggered when the protections change while OPSTEP > 0x0

0x1B: MODE in PKA_CR does not match the expected value (selected by CCOP in CCB_CR) when OPSTEP > 0x0.

0x1C: OPSTEP end-step event not found at OPSTEP= 0x0 (SAES KEYVALID is set). Use CCB IPRST before starting a coupling and chaining operation.

0x1D: OPSTEP end-step event not found at OPSTEP = 0x1.

0x1E: OPSTEP end-step event not found at OPSTEP = 0x2.

0x1F: OPSTEP end-step event not found at OPSTEP = 0x3.

0x20: OPSTEP end-step event not found at OPSTEP = 0xA.

0x21: OPSTEP end-step event not found at OPSTEP = 0x12.

0x22: OPSTEP end-step event not found at OPSTEP = 0x13.

0x23: OPSTEP end-step event not found at OPSTEP = 0x17.

For error codes 0x1C to 0x23, details on end-step events are found in *Section 29.5*. Those errors are also triggered when values written in SAES_CR register do not correspond to the selected CCOP and the expectations in the current OPSTEP.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **OPSTEP[4:0]**: Operation step

This read-only information indicates the current step of the coupling and chaining operation (indicated by CCOP). See *Section 29.5* for details.

This bitfield can be cleared only using IPRST in CCB_CR. OPSTEP[4:0] = 0x1F when the IPRST procedure is in progress.

29.7.3 CCB reference tag register (CCB_REFTAGRx)

Address offset: 0x010 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

Writes to this register are ignored when CCB_BUSY is set in the CCB_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFTAG{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REFTAG{i + 32 * x}**: Reference tag bits {i + 32 * x} (i = 31 to 0)

The application must write in these registers, the reference GCM tag of the required operation blob (respectively ECC signature, ECC key, or RSA key). This enables the PKA protected operation (respectively ECC signature/public key computation, scalar multiplication, or modular exponentiation). Refer to *Section 29.5* for details.

Write to this register is ignored unless OPSTEP = 0x12 in CCB_SR.

This register is cleared when IPRST is set and CCB_BUSY is cleared.

29.7.4 CCB register map

Table 258. CCB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CCB_CR	IPRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCOP[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0x004	CCB_SR	Res.	Res.	TAMP_EVT5	TAMP_EVT4	TAMP_EVT3	TAMP_EVT2	TAMP_EVT1	TAMP_EVT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCB_BUSY	Res.	Res.	OPERR[5:0]						Res.	Res.	Res.	OPSTEP[4:0]				
	Reset value			0	0	0	0	0	0								0			0	0	0	0	0	0				0	0	0	0	0
0x010 + 0x4 * x (x=0 to 3)	CCB_ REFTAGRx	REFTAG{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

30 True random number generator (RNG)

30.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP 800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

30.2 RNG main features

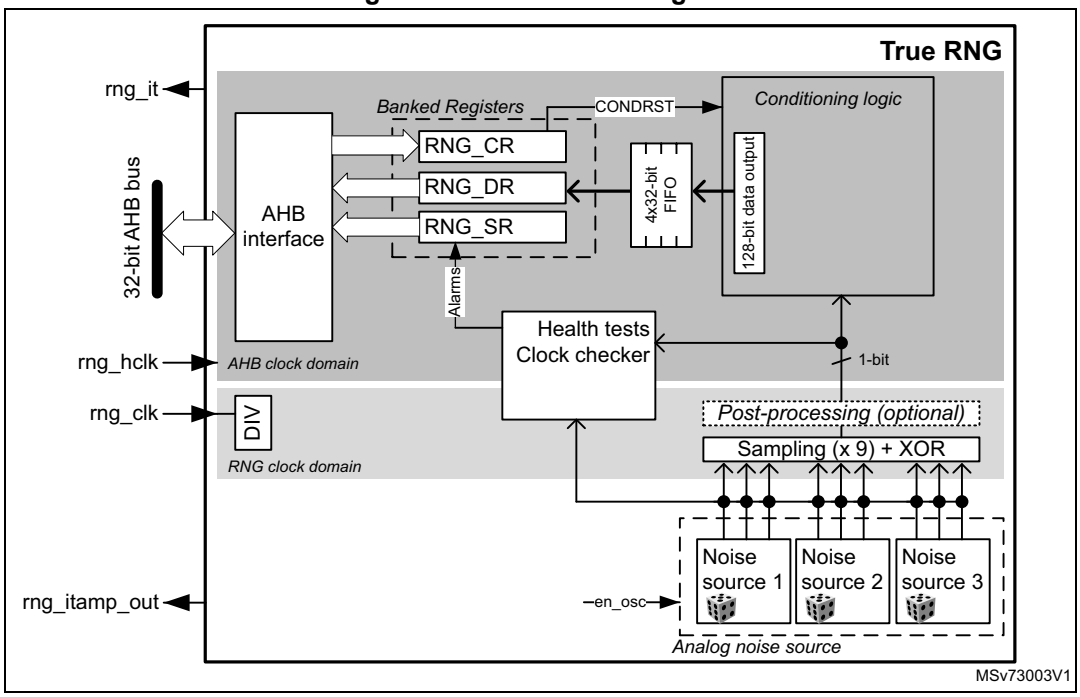
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by an NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the NIST configuration, produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- Embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- Can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- Has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

30.3 RNG functional description

30.3.1 RNG block diagram

Figure 251 shows the RNG block diagram.

Figure 251. RNG block diagram



30.3.2 RNG internal signals

Table 259 describes internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 259. RNG internal input/output signals

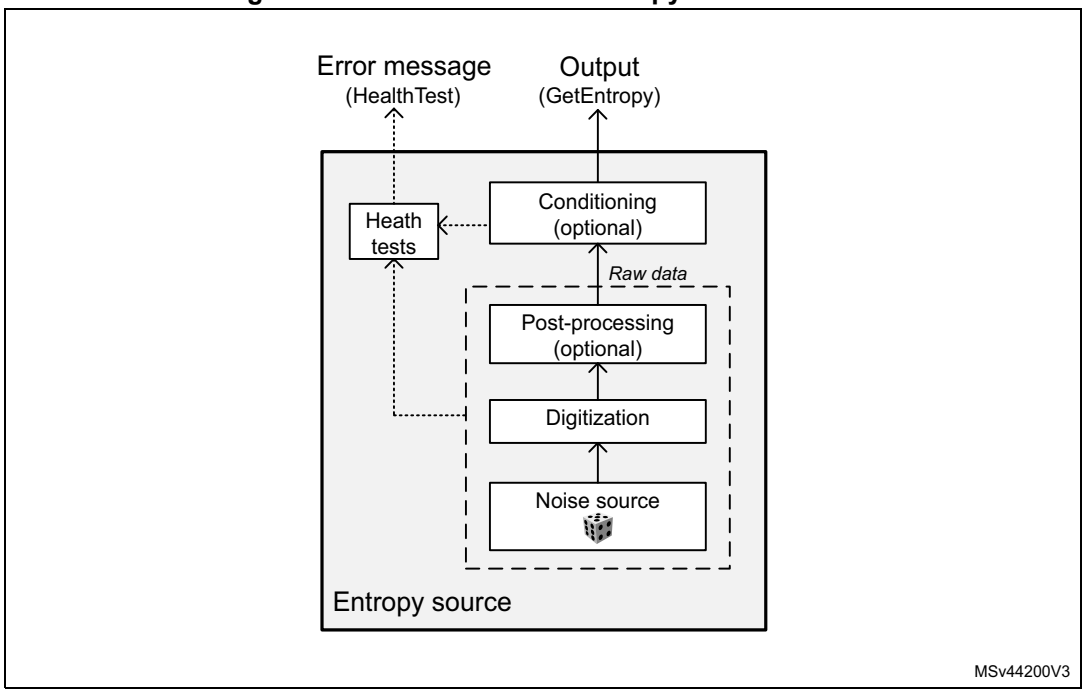
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples. A reset and a new initialization are needed to use it again.

30.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary, RNG integrates all the required NIST components shown in *Figure 252*: an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block, and two interfaces used to interact with the entropy source (GetEntropy and HealthTest).

Figure 252. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter.

Noise source

This component contains the nondeterministic, entropy-providing activity ultimately responsible for the uncertainty associated with the bit string output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x3), each based on three XOR-ed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 30.3.8*. Multiple oscillators are also disabled for configuration A (see *Table 261*).
- The XOR-ing of all the noise sources into a single analog output.
- A sampling stage of this output is clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent from the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: The recommended clock frequencies and associated divider value are detailed in Section 30.6.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In the non-NIST configuration B (as defined in *Section 30.6.2*) a normalization debiasing is applied: half of the bits are taken from the sampled noise source, the other half are taken from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bit string output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The time intervals required between two random number generations, and between the RNG initialization and availability of the first sample are detailed in *Section 30.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 30.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When the interrupts are enabled, an interrupt is generated when this flag transitions from 0 to 1. The interrupt is then cleared automatically by the RNG, as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts, then operates as expected, obtaining the assurance that failures are caught quickly, with high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCRx (configuration A in *Section 30.6.2*). The description below refers to the RNG_HTCR0 configuration.

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 40 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 688 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous test:
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than the AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 30.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 30.3.7*.

Note: An interrupt can be generated when an error is detected.

The health test thresholds are modified by changing the value in the corresponding RNG_HTCR register. See Section 30.6 for details.

30.3.4 RNG initialization

The RNG simplified state machine is shown in *Figure 253*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering the startup behavior test on the raw data samples and known-answer tests.
3. Startup health tests are completed. During this time, three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 30.5*.

Figure 253. RNG initialization overview

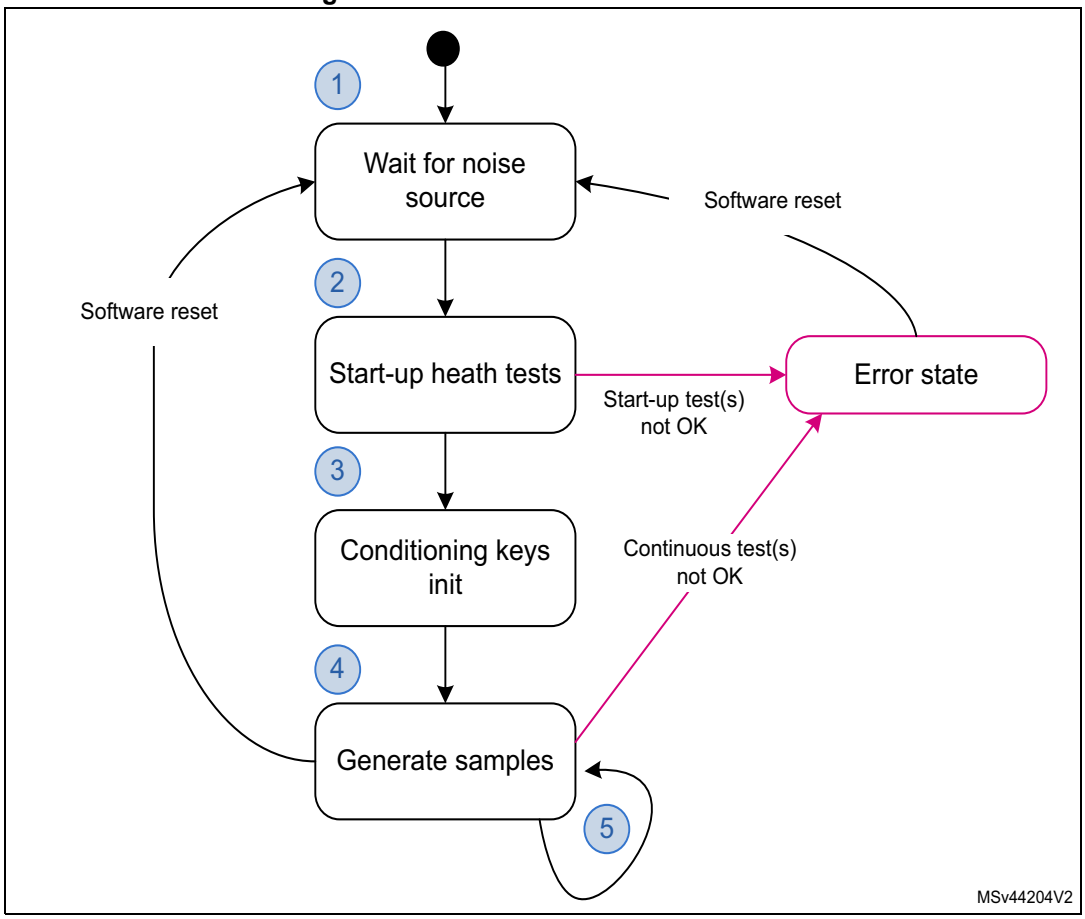


Figure 253 also highlights a possible software reset sequence, implemented by:

1. Write RNGEN = 0, then wait for BUSY = 0 in the RNG_SR register.
2. Write bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
3. Write RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
4. Wait for a random number to be ready, after initialization completes.

Note: When the RNG is reset by RCC (hardware reset), the configuration for optimal randomness is lost in the registers. Software reset with CONFIGLOCK set preserves the configuration.

30.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check *Section 30.6* and verify if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY=0, write in the RNG_CR register bit CONDRST = 1, together with the correct RNG configuration. Then perform a second write with bit CONDRST = 0, interrupt enable bit IE = 1, and RNG enable bit RNGEN = 1.
   - If this is not the case, perform a write to the RNG_CR register with interrupt enable bit IE = 1 and RNG enable bit RNGEN = 1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data are available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 30.3.7* must be used.

To run the RNG in polling mode the following steps are recommended:

1. Check *Section 30.6* and verify if a specific RNG configuration is required for the application.
   - If this is the case, when bits RNGEN and BUSY = 0, write in the RNG_CR register bit CONDRST = 1, together with the correction RNG configuration. Then perform a second write to the RNG_CR register with bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If this is not the case, only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)

3. If the above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data are available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 30.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns 0. It is recommended to always verify that RNG_DR is different from 0. When this happens, a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required, it is possible to select a faster configuration by using the RNG configuration “B”, described in *Section 30.6*. The gain in random number generation speed is summarized in *Section 30.5*.

Low-power operation

If power consumption is a concern, low-power strategies can be used, as described in *Section 30.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 30.3.3*.

30.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 30.6*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 30.3.1* for details (AHB and RNG clock domains).

30.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 30.3.6*), and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

Note: The clock error has no impact on generated random numbers: the application can still read the RNG_DR register.

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

When ARDIS is cleared (auto-reset enabled), clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.

When ARDIS is set (auto-reset disabled), the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the autoreset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG, an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral, using the central reset management or the global SoC reset. A proper initialization of the RNG is required again.

30.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while the RNGEN = 0, the following features are available to the software:

- If there are valid words in the output buffer, four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register, four additional random numbers can be still be read from the RNG_DR register. If it is not the case, RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode), and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 30.5*.

When disabling the RNG, the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when the RNGEN bit is set to 1, unless the application resets the conditioning logic using the CONDRST bit in the RNG_CR register.

When the application wants to gate, the RNG clock waits for BUSY bit to be cleared in the RNG_SR register after clearing the RNGEN bit and before gating the RNG kernel clock using the RCC. Similarly, when an application needs to enter a power mode where RNG is deactivated, wait until BUSY bit is cleared in the RNG_SR register after clearing the RNGEN bit and before entering the low power mode using the PWR.

Note: The power modes where RNG is deactivated (retained or not available) can be found in the PWR section.

30.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 30.3.7*
- Clock error, see *Section 30.3.7*

Dedicated interrupt enable control bits are available as shown in *Table 260*.

Table 260. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 30.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

30.5 RNG processing time

In the recommended configuration A or C described in *Table 261*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz. The value N is defined in *Section 30.6*.

Note: When CLKDIV is different from 0, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 206 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

30.6 RNG entropy source validation

30.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

30.6.2 Validation conditions

The RNG true random number generator has been tested in the following conditions:

- RNG clock rng_clk= 48 MHz, with CED bit cleared in RNG_CR.
- RNG configurations are described in *Table 261*. Note that only configuration A can be certified NIST SP800-90B. Refer to *Table 263* to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 261. RNG configurations

RNG Configuration	RNG_CR bits					Loop number (N)	RNG_ HTCR0 register	RNG_ NCSR register
	NISTC bit	RNG_ CONFIG1 [7:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0]	RNG_ CONFIG3 [3:0](1)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program.*							
B	1	0x01	0x0(2)	0x0	0x1	1	0x0000 AAC7(3)	Default
C	0	0x82		0x0	0xF	2		Default

1. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N = 1, 0x1 to N = 2, 0x2 to N = 3, and 0x3 to N = 4.
2. The noise source sampling must be 48 MHz or less. If the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 30.7.1* for details.
3. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

Table 262. Additional health test configurations

RNG configuration	RNG_HTCR1 register (oscillator set 1)	RNG_HTCR2 register (oscillator set 2)	RNG_HTCR3 register (oscillator set 3)
A	default	default	default
B and C	default	default	default

Table 263. Configuration selection

Section criteria	Configuration A	Configuration B	Configuration C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using the AIS-31 test suite (T0 to T8).
2. When the speed is not enough for the application, a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*Introduction to random number generation validation using the NIST statistical test suite for STM32 MCUs and MPUs*", available on www.st.com.

30.7 RNG registers

The RNG is associated with a control register, a data register, and a status register.

30.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG configuration lock

0: Writes to the RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and the RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_NSCR, RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that sets any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

Before setting this bit to 1 RNGEN must be set to 0, and BUSY flag must be cleared in RNG_SR.

When CONDRST is set to 0, wait for BUSY flag to be cleared in RNG_SR. CONDRST bit then reads as 0.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 30.6*.

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing to this bitfield is only effective when CONFIGLOCK = 0. The new value is only taken into account if CONDRST bit is set in the same access.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration for 128-bit output, two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 30.6* for proposed configuration.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset is enabled

1: Auto-reset is disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 30.3.7*.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection is enabled.

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing to this bit is only effective when CONFIGLOCK = 0. The new value is only taken into account if the CONDRST bit is set in the same access.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS flag is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

This bit enables/disables the true random number generator.

0: True random number generator disabled.

1: True random number generator enabled.

Clearing RNGEN powers off the analog noise sources and gates the logic clocked by the RNG clock.

When clearing this bit verify that BUSY flag is cleared in RNG_SR before updating this register again.

Bits 1:0 Reserved, must be kept at reset value.

30.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0	r		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 4 **BUSY:** Busy

This flag indicates whether RNG is idle or busy.

0: Idle

1: Busy

RNG is flagged as busy when a conditioning soft reset (using CONDRST) is in progress. It is also flagged busy when RNG is activated by application setting RNGEN bit, or when a connected peripheral fetches randoms via the dedicated RNG hardware bus.

More information on BUSY usage can also be found in *Section 30.3.8*.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

– Start-up or continuous repetition count test on noise source failed.

– Start-up or continuous adaptive proportion test on noise source failed.

– Start-up post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

This flag is raised to indicate valid (non-0) random data available in the RNG_DR register. It is cleared by hardware upon reading the RNG_DR register (which empties the RNG buffer). The DRDY flag can also be raised upon disabling RNG (upon clearing the RNGEN bit).

0: No valid random data ready.

1: Valid random data ready.

Raising DRDY generates an interrupt if enabled with the IE bit of the RNG_CR register.

30.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

This register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, valid when DRDY = 1. When DRDY = 0, the RNDATA value is 0.

The 0 value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

30.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0000 01FF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

30.7.5 RNG health test control register x (RNG_HTCRx)

Address offset: 0x010 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 72AC, 0x0003 FFFF, 0x0003 FFFF, 0x0003 FFFF

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests (repetition count test, adaptive proportion test) at position x. See *Section 30.6* for the recommended value. With its default value, health tests have a very small probability of triggering any error at position x.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

30.7.6 RNG register map

Table 264. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	BUSY	Res.	SECS	CECS	DRDY
	Reset value																										0	0	0		0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
	Reset value																								1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR0	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0
0x014	RNG_HTCR1	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	RNG_HTCR2	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01F	RNG_HTCR3	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

31 AES hardware accelerator (AES)

31.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

31.2 AES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

31.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 265. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

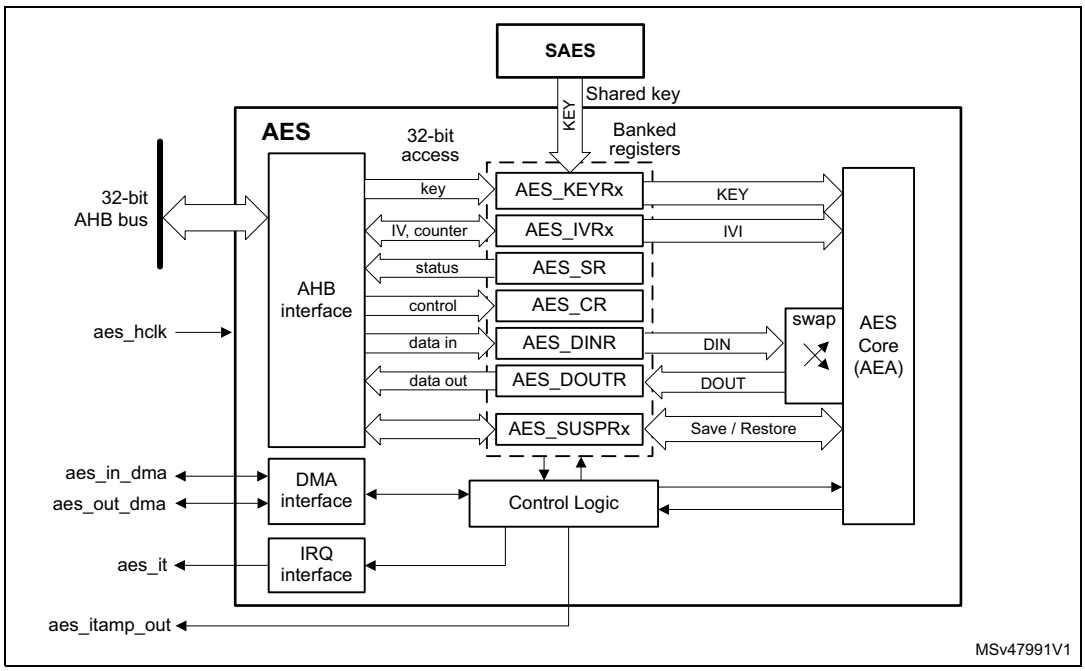
1. X = supported.

31.4 AES functional description

31.4.1 AES block diagram

Figure 254 shows the block diagram of AES.

Figure 254. AES block diagram



31.4.2 AES internal signals

Table 266 describes the user relevant internal signals interfacing the AES peripheral.

Table 266. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	AES incoming data DMA single request/acknowledge
aes_out_dma	Input/Output	AES processed data DMA single request/acknowledge
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

31.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

31.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 267*. Those functions can be certified NIST PUB 140-3.

Table 267. AES approved symmetric key functions

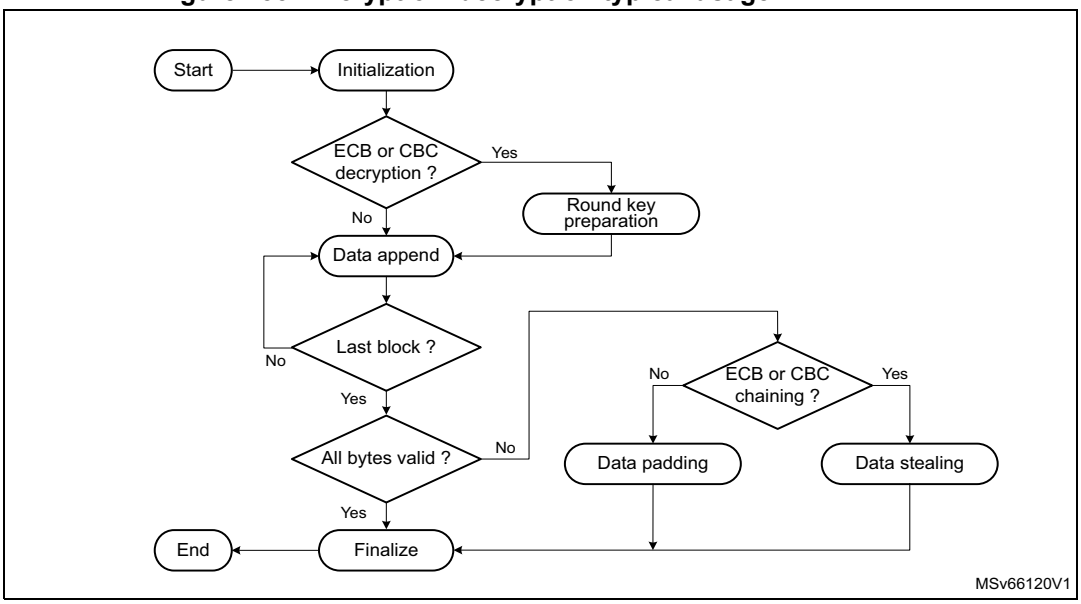
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 31.4.8*.

31.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 255. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 31.4.9: AES basic chaining modes (ECB, CBC)* and *Section 31.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 31.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 31.6: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.

3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 31.4.8* is not supported.

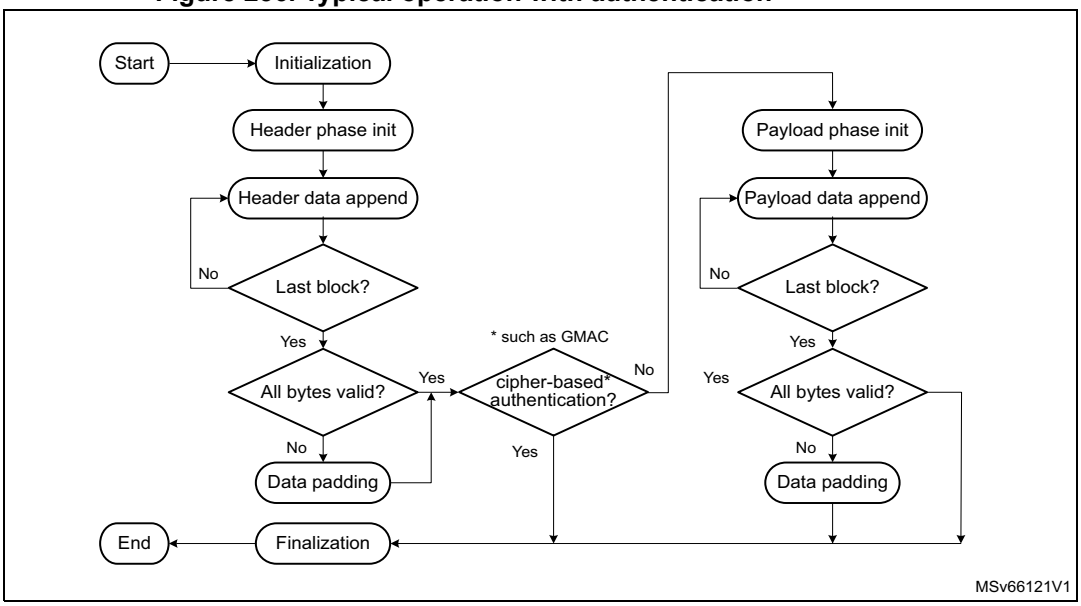
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

31.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 256. Typical operation with authentication



Section 31.4.11: AES Galois/counter mode (GCM) and *Section 31.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 31.4.12: AES Galois message authentication code (GMAC)*.

31.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques

such as those described in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since AES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

31.4.8 AES suspend and resume operations

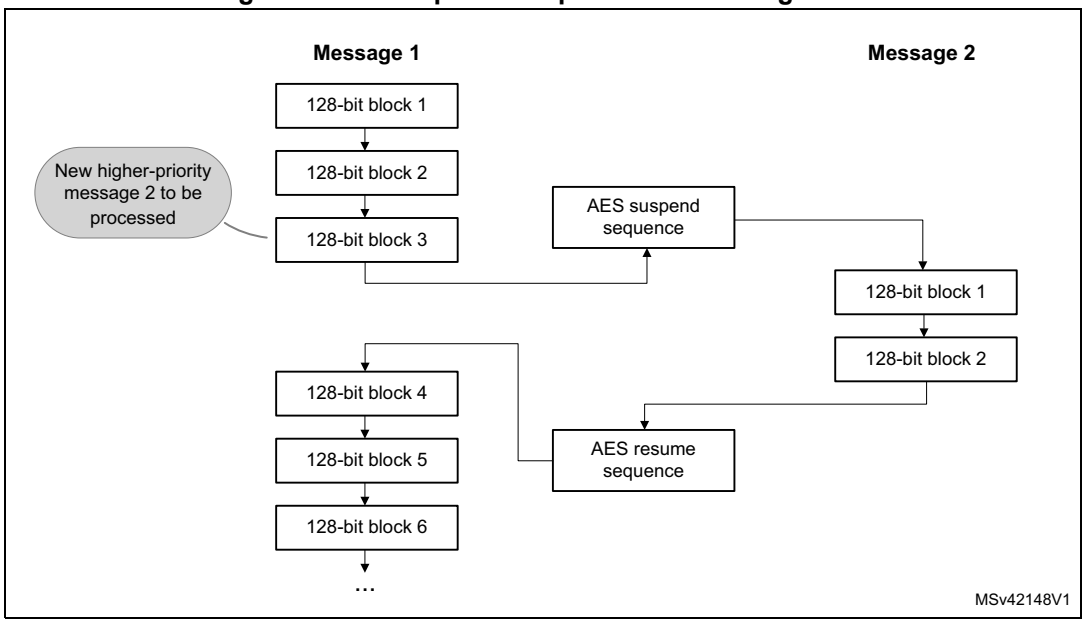
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 257 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 257. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

31.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 258 and *Figure 259* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 258. ECB encryption

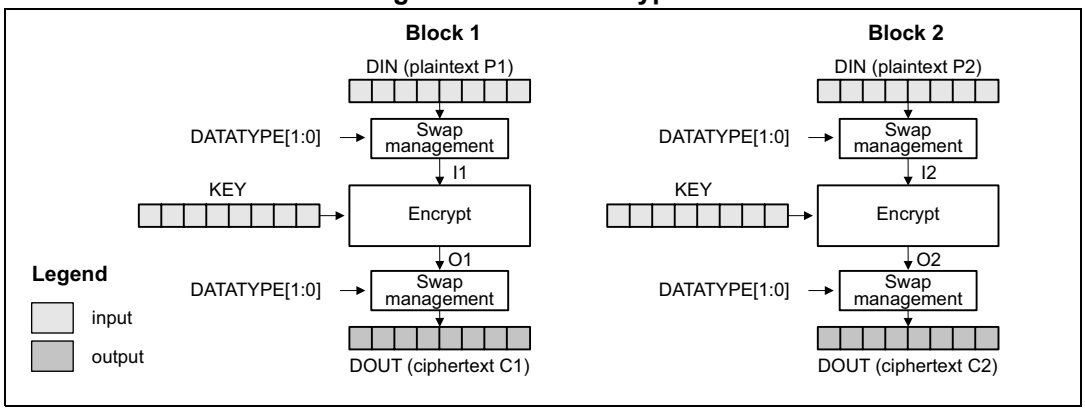
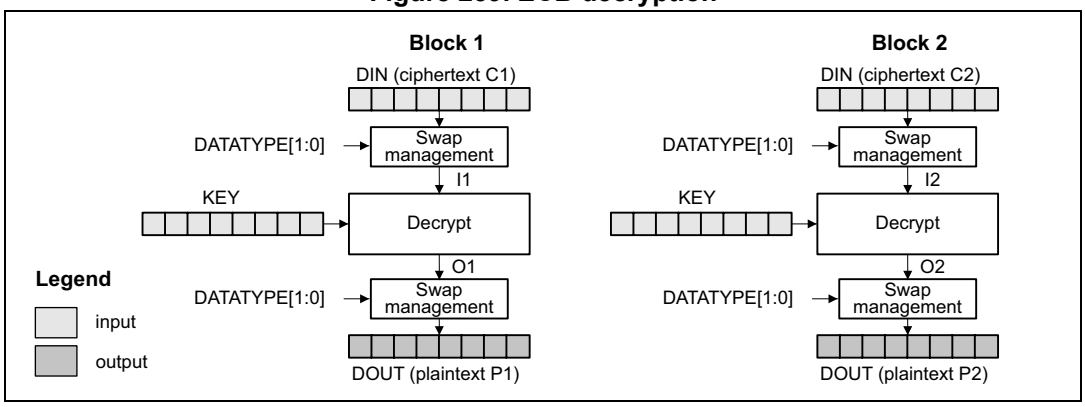


Figure 259. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 260 and *Figure 261* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 260. CBC encryption

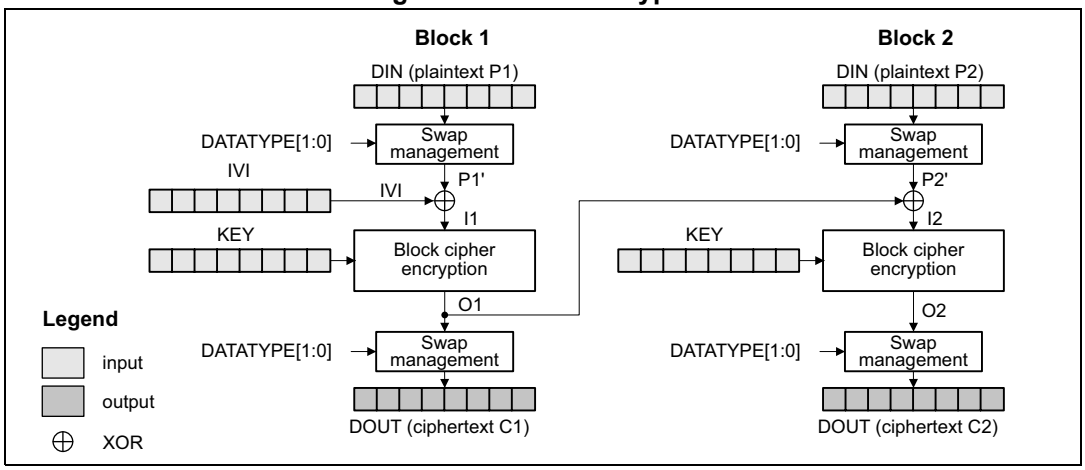
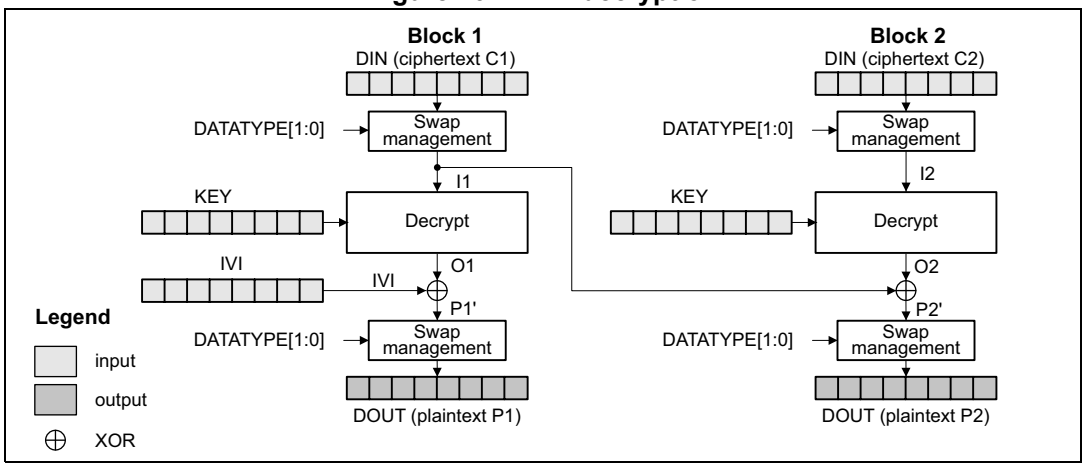


Figure 261. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 31.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.

3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 31.4.7*.
   b) Append the cleartext block into AES as described in *Section 31.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 31.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   – Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE.
   – Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.

11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 31.4.7*.
    b) Append the ciphertext block into AES as described in *Section 31.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).
5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 31.4.14*).
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

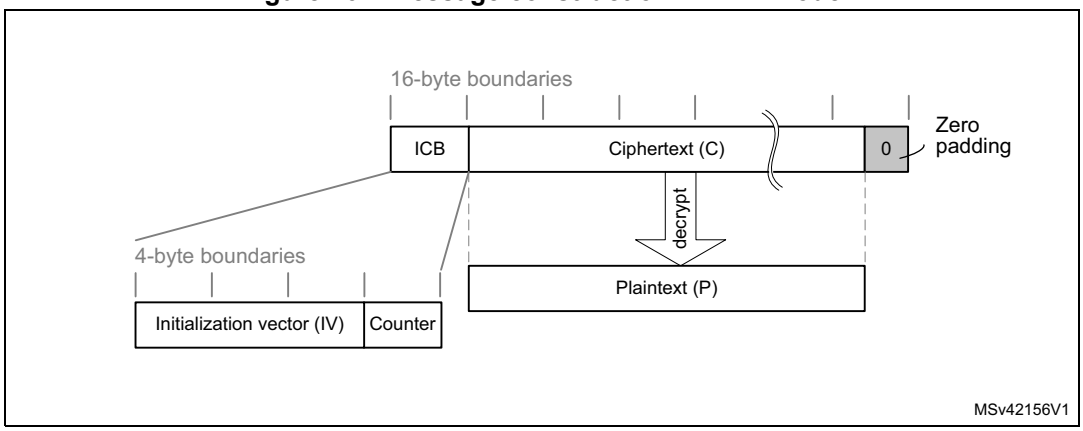
Note: It is not required to save the key registers as the application knows the original key.

31.4.10 AES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 262*.

Figure 262. Message construction in CTR mode



The structure of this message is:

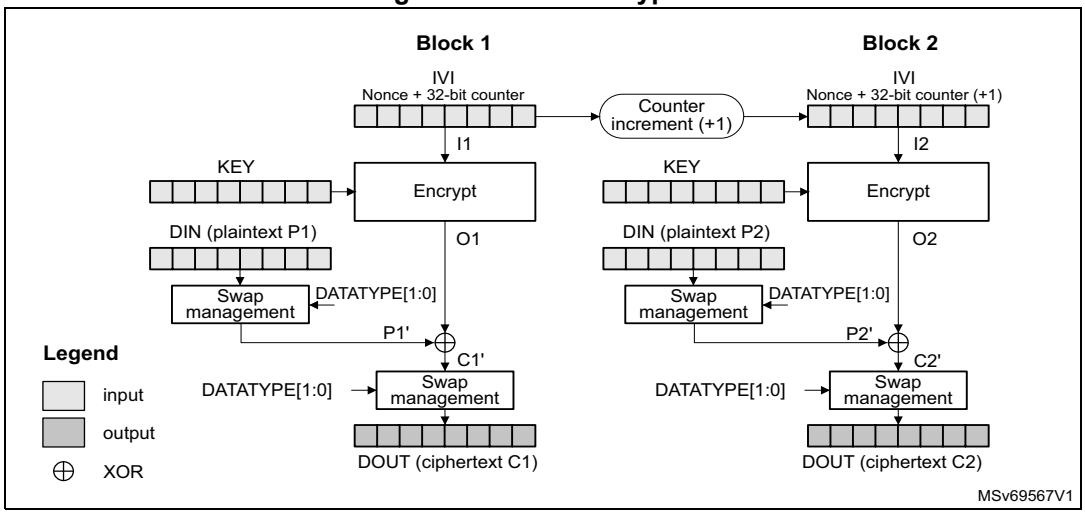
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 263 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 263. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 268*.

Table 268. Counter mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 31.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to *Table 268*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 31.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 31.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

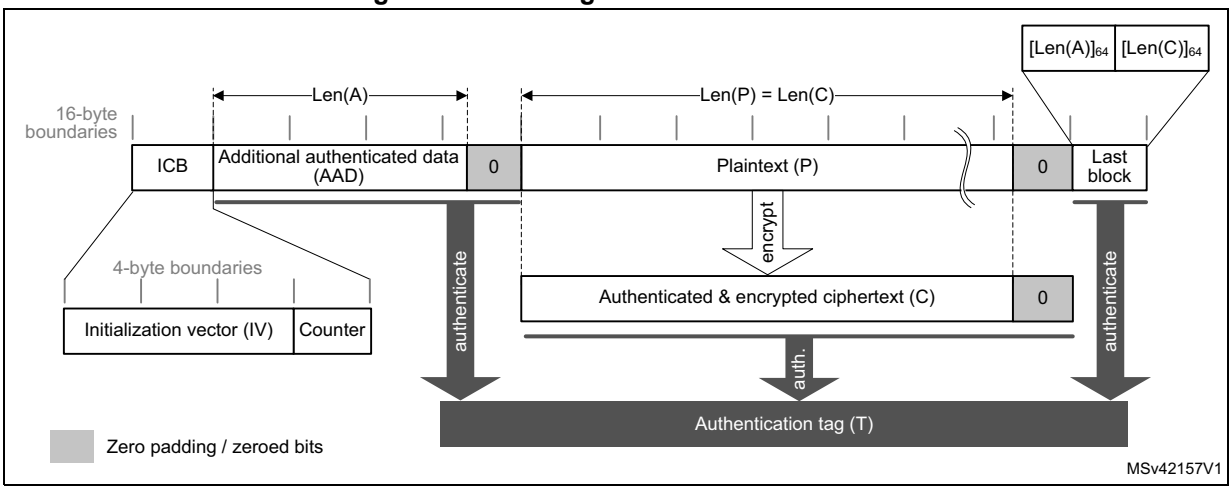
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

31.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 264. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 270*.

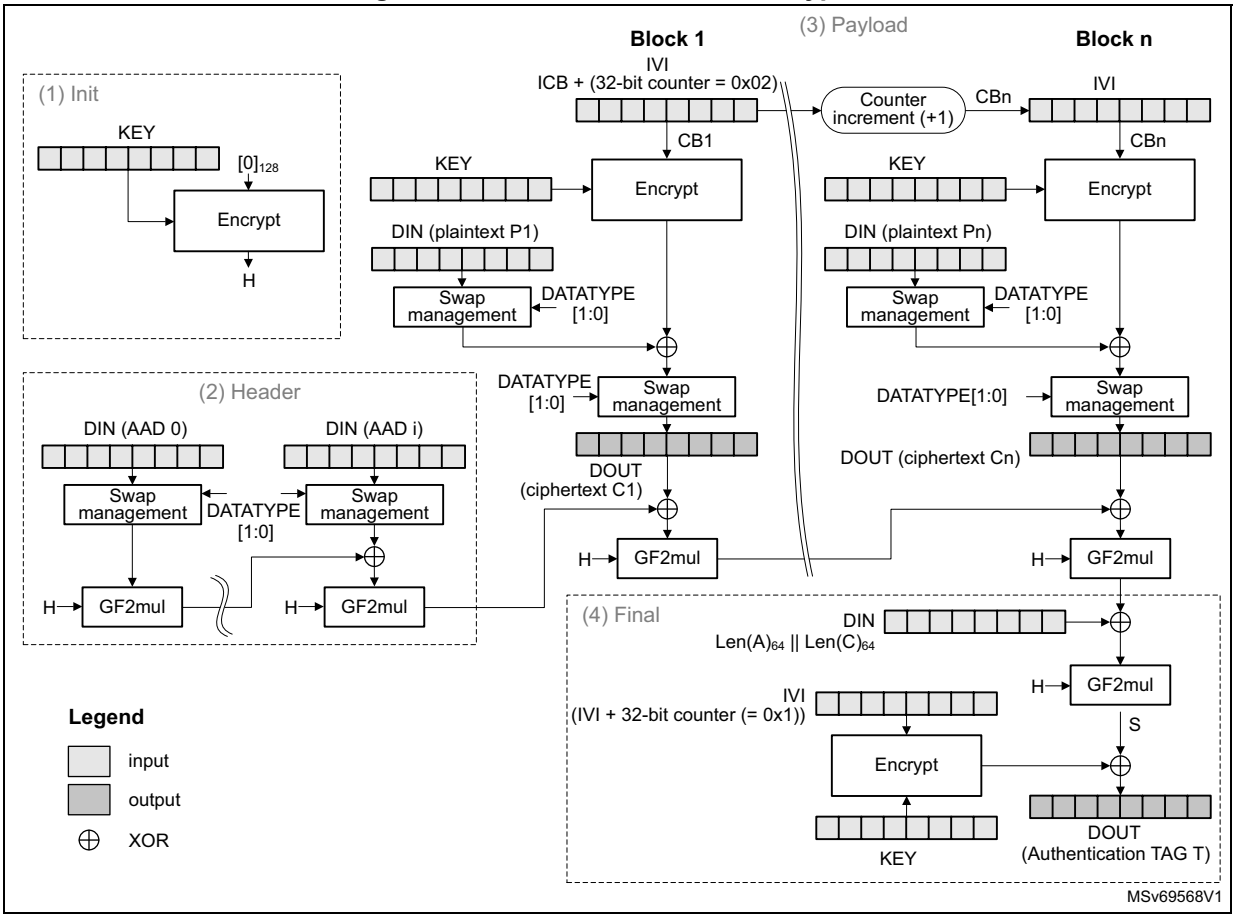
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 265 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 265. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 269*.

Table 269. Initialization of IV registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 270*.

Table 270. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 31.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 269*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 31.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 31.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 270*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

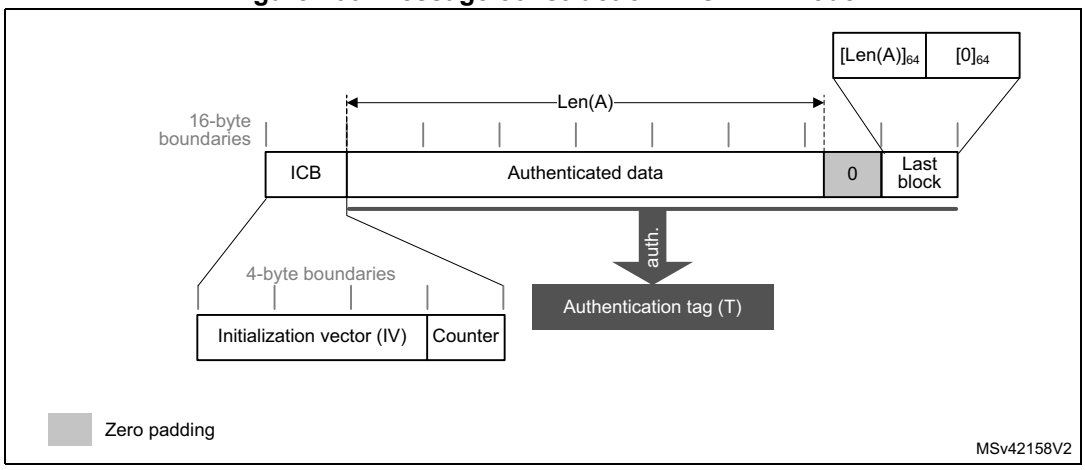
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
5. Enable the AES peripheral, by setting EN.

31.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

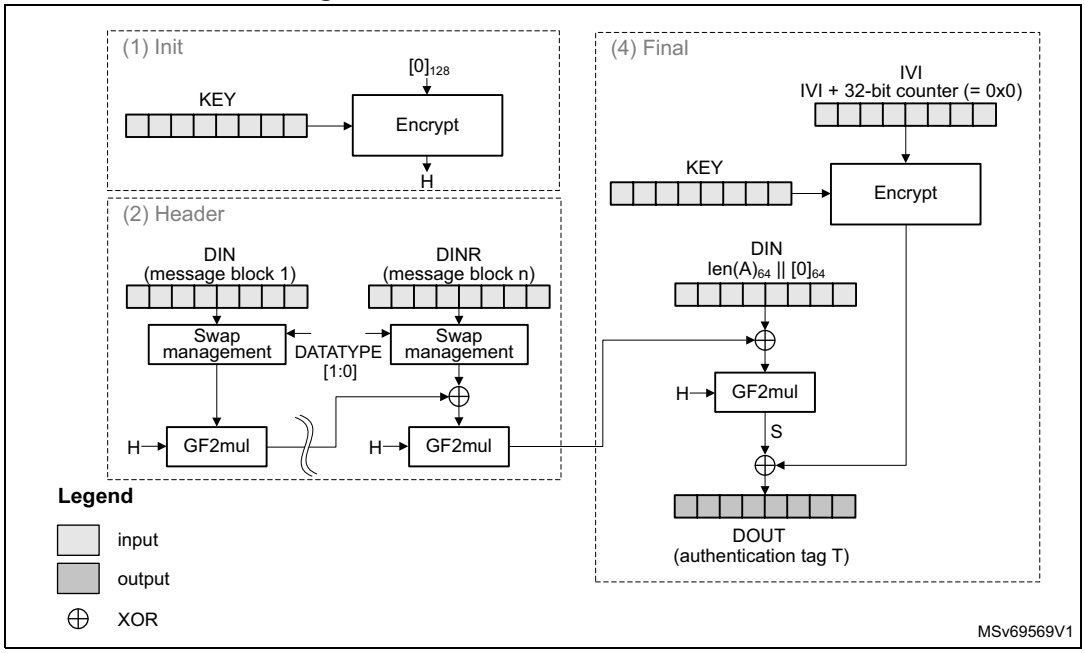
Figure 266. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 267 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 267. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

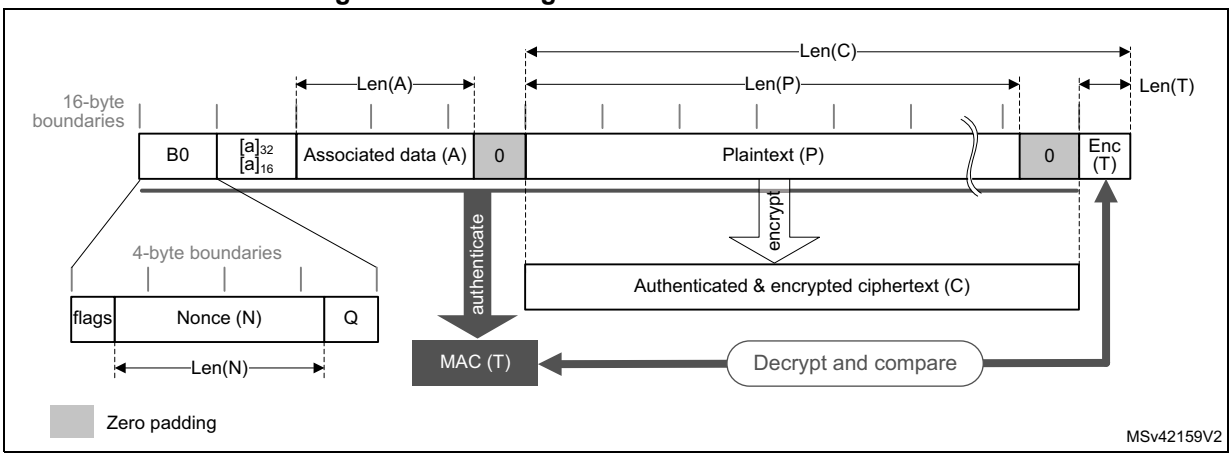
31.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 268. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 268*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 268*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

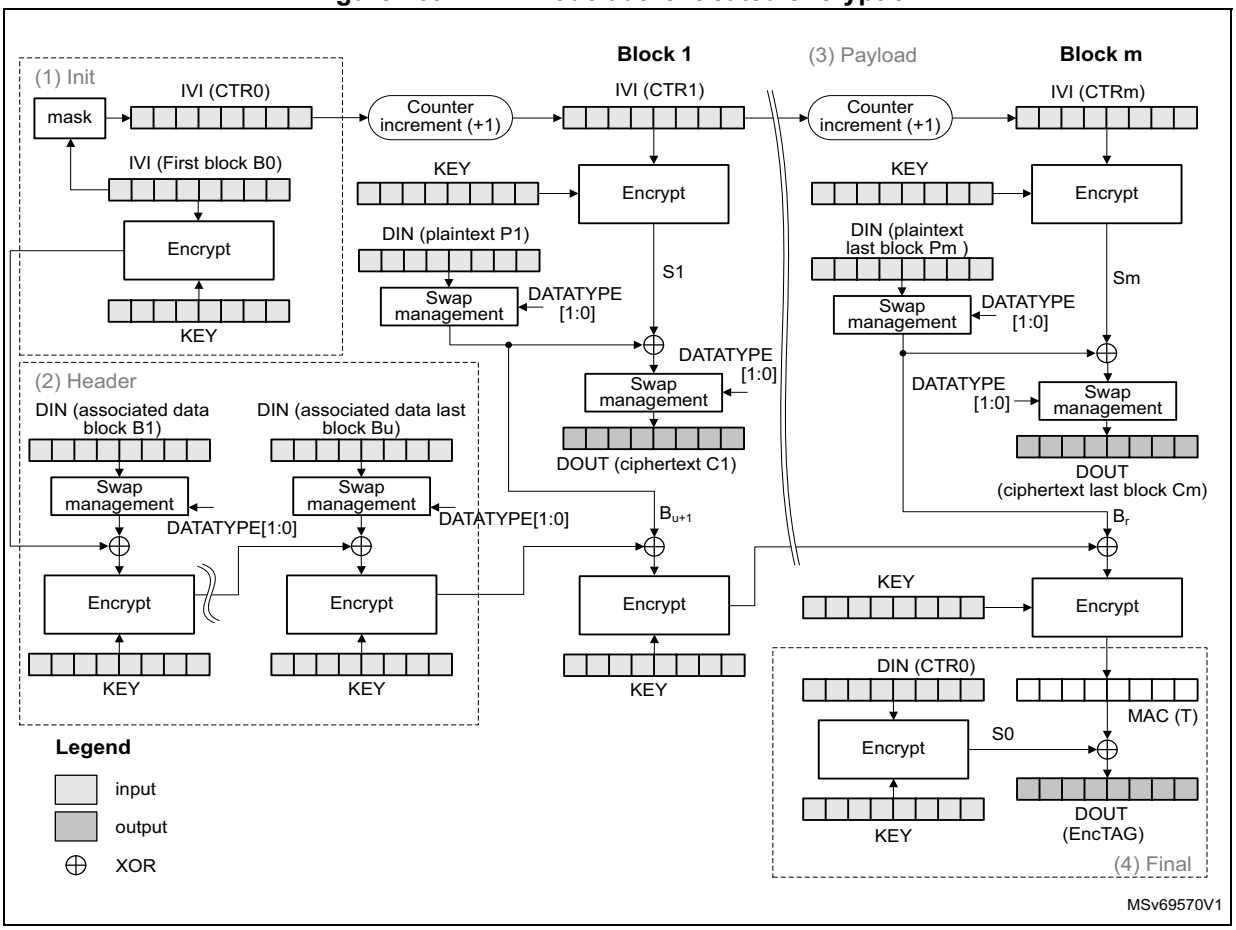
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 269 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 269. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 271*.

Table 271. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 31.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 271*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 31.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 31.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 31.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 31.4.14*).
5. Enable the AES peripheral, by setting EN.

31.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section *SAES operations with shared keys* in the *SAES* section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

31.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 272* “no swapping” option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 272* and *Figure 270*.

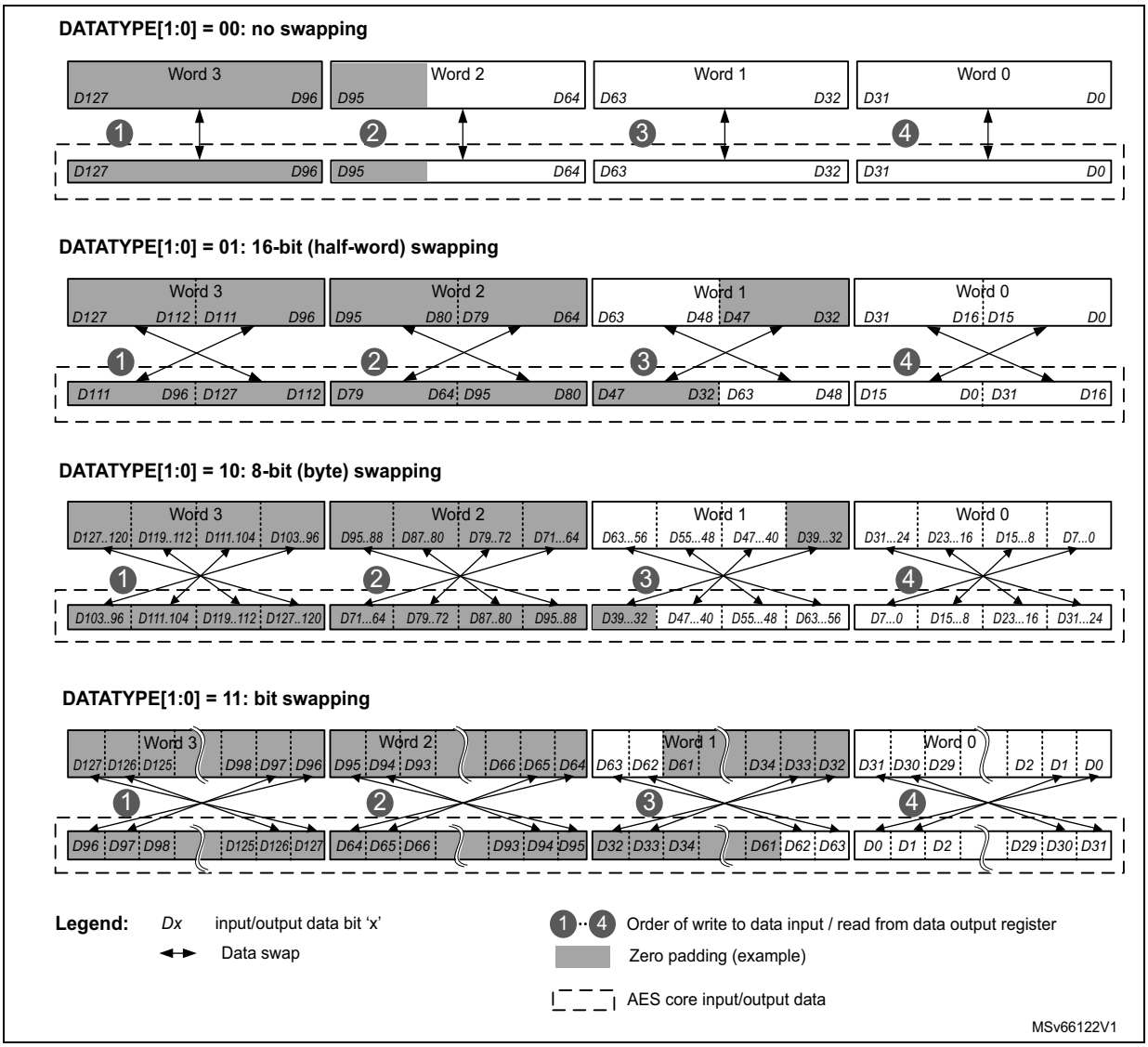
Table 272. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

Table 272. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 270. 128-bit block construction according to the data type



Data padding

Figure 270 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

31.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 273*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 273. Key endianness in AES_KEYRx registers (128/256-bit keys)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 31.4.18*).

31.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 274*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 274. IVI bitfield spread over AES_IVRx registers

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

31.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 31.5: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 31.5: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 31.4.16: AES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 31.4.14: AES key sharing with secure AES co-processor* for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 31.5: AES interrupts*.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring AES.

31.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 31.4.18: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 275 gives a summary of the available features.

Table 275. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

31.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 31.4.5: AES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 31.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

31.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 276. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 277. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

31.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

31.8.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw						rw	rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the AES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable.

0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection
This bitfield defines the key length in bits of the key used by AES.
0: 128-bit
1: 256-bit
Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 31.4.15: AES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the AES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

31.8.2 AES status register (AES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 31.4.16: AES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy.
0: Idle
1: Busy
AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.
AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.
When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.
0: No error
1: Unexpected write to AES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.
0: No error
1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

31.8.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.

31.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

31.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

31.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_CR register

31.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

31.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

31.8.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

31.8.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

31.8.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 31.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

31.8.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

31.8.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

KEIF is raised upon any of the following events:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).

–AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

31.8.20 AES interrupt clear register (AES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the AES_ISR register.

31.8.21 AES register map

Table 278. AES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1]	KMOD[0]	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0						0	0	0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 278. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Table 278. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	AES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x309-0x3FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

32 Secure AES coprocessor (SAES)

32.1 Introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (wrapped-key mode) and key sharing with faster AES peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

32.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)

- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

32.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.

Table 279. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

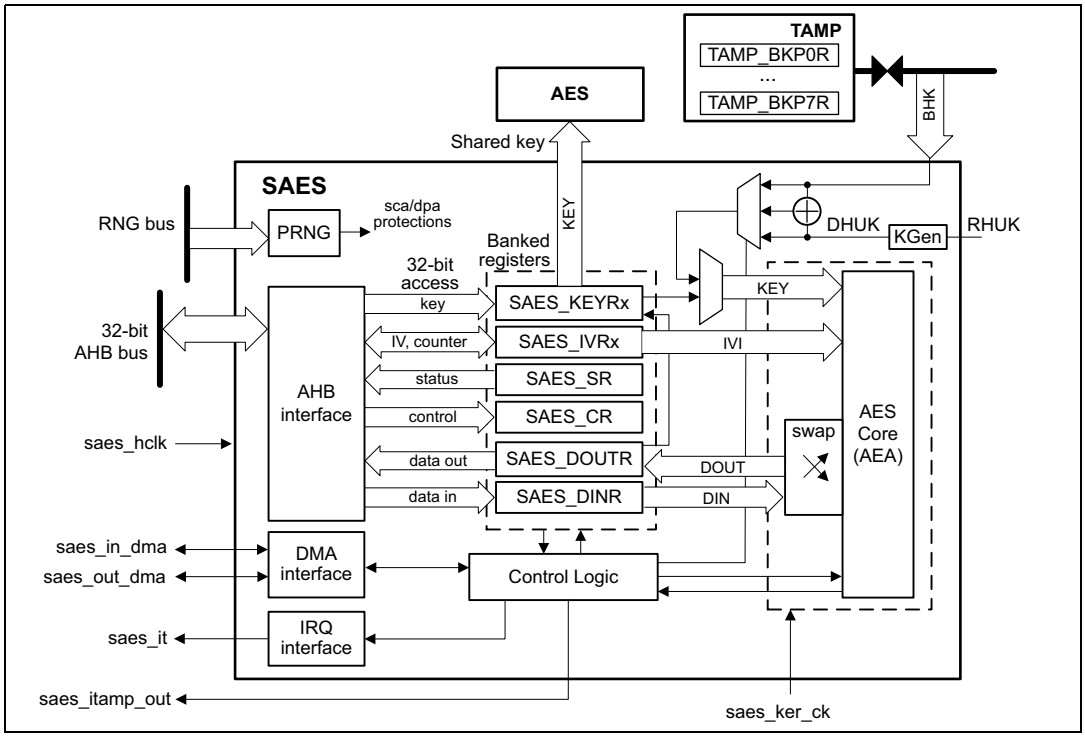
1. X = supported.

32.4 SAES functional description

32.4.1 SAES block diagram

Figure 271 shows the block diagram of SAES.

Figure 271. SAES block diagram



Note: AES represents the AES peripheral.

32.4.2 SAES internal signals

Table 280 describes the user relevant internal signals interfacing the SAES peripheral.

Table 280. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
RHUK	Input	256-bit root hardware unique key (non-volatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 32.4.17* for details).

32.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 32.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 32.4.19* for details.

This background task must be completed before the device enters a low-power mode.

32.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 281*. Those functions can be certified NIST PUB 140-3.

Table 281. SAES approved symmetric key functions

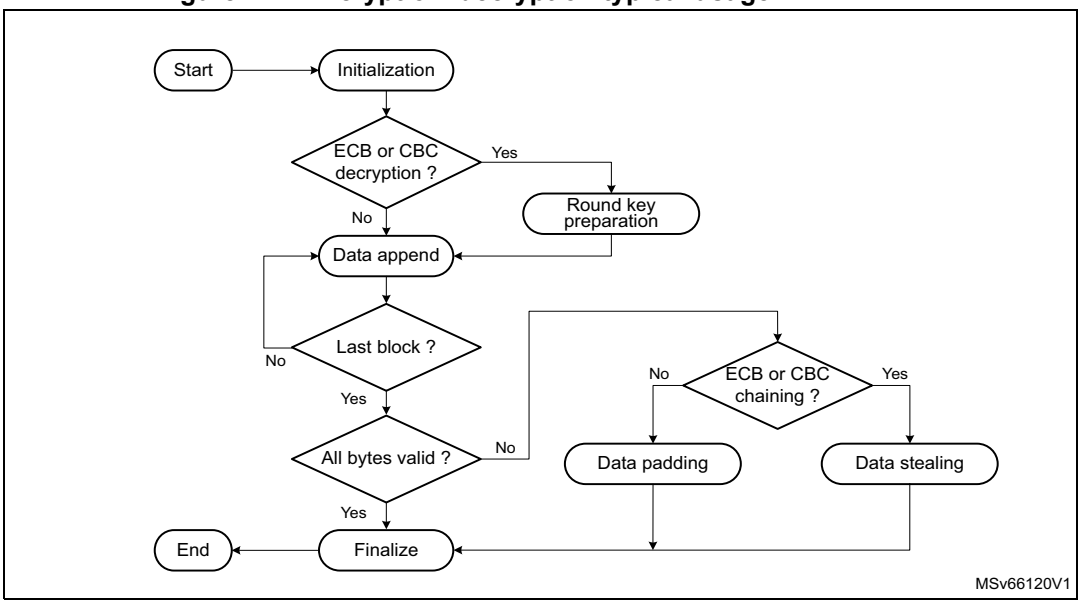
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 32.4.8*.

32.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 272. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 32.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 32.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 32.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 32.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 32.4.8* is not supported.

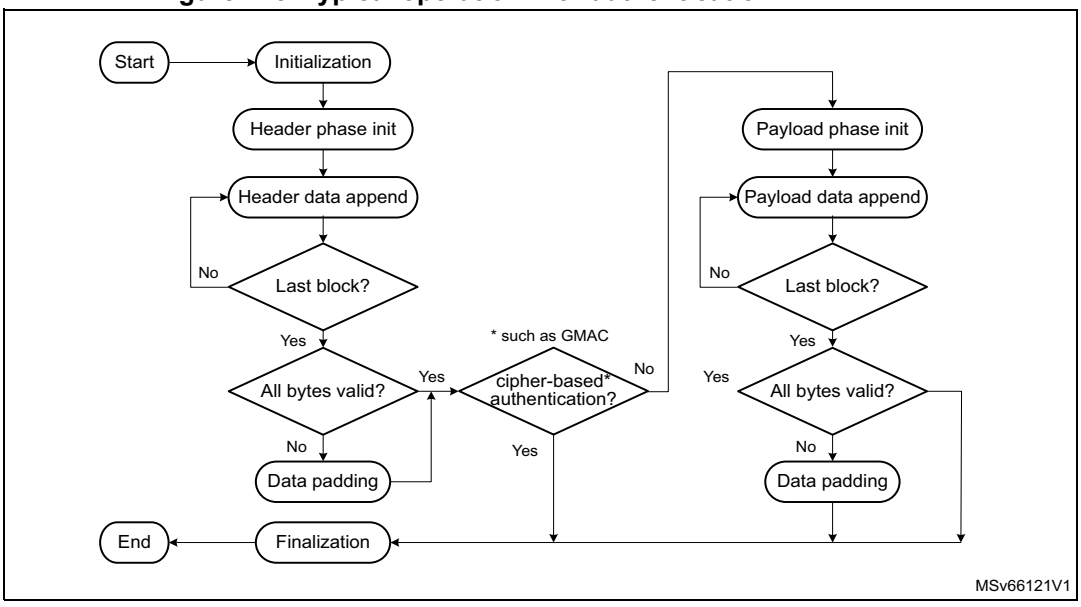
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

32.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 273. Typical operation with authentication



Section 32.4.11: SAES Galois/counter mode (GCM) and *Section 32.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 32.4.12: SAES Galois message authentication code (GMAC)*.

32.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A*, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

32.4.8 SAES suspend and resume operations

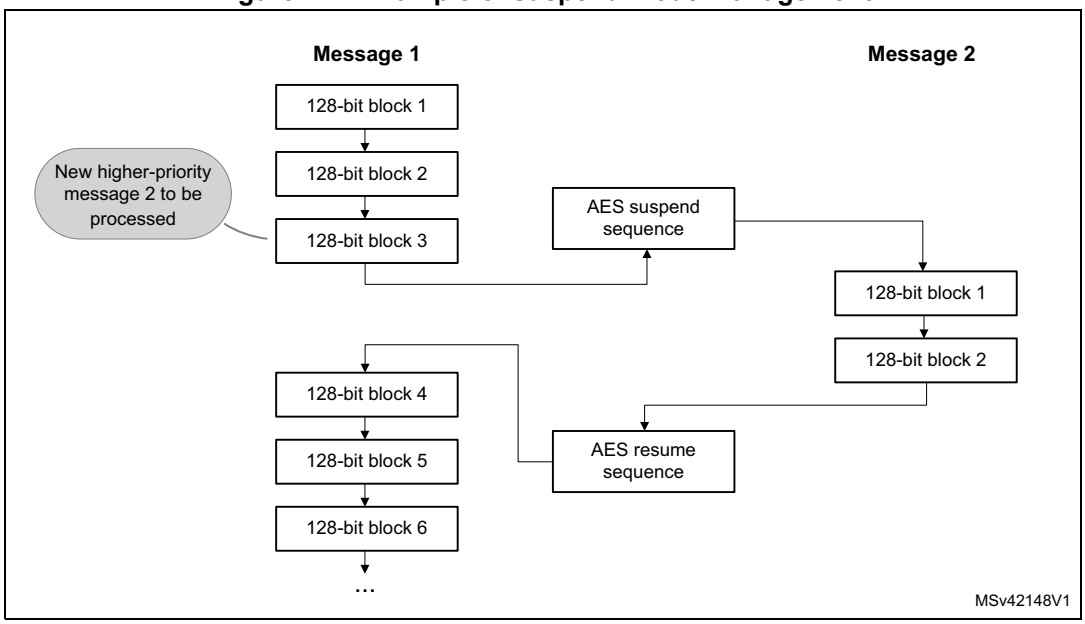
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 274 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 274. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

32.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 275 and *Figure 276* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 275. ECB encryption

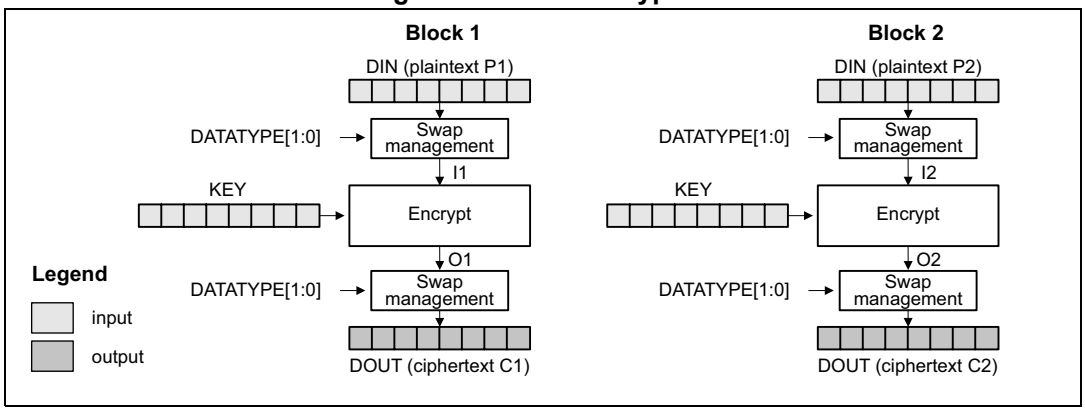
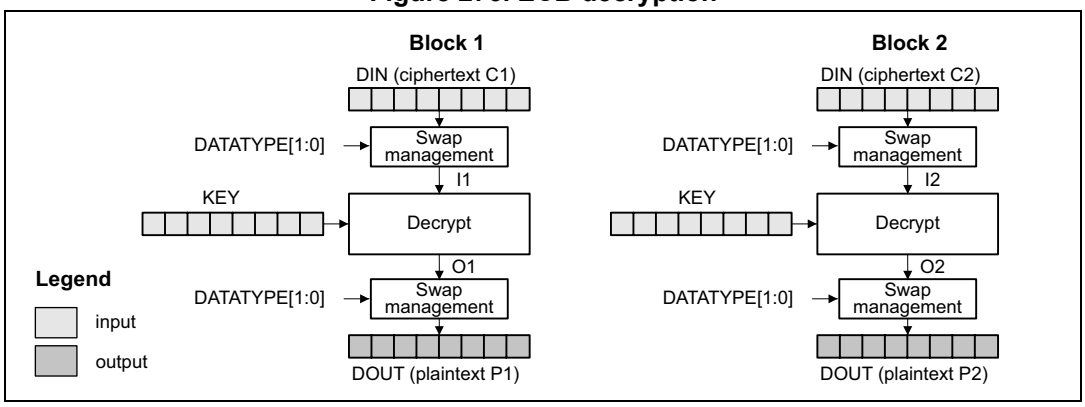


Figure 276. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 277 and *Figure 278* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 277. CBC encryption

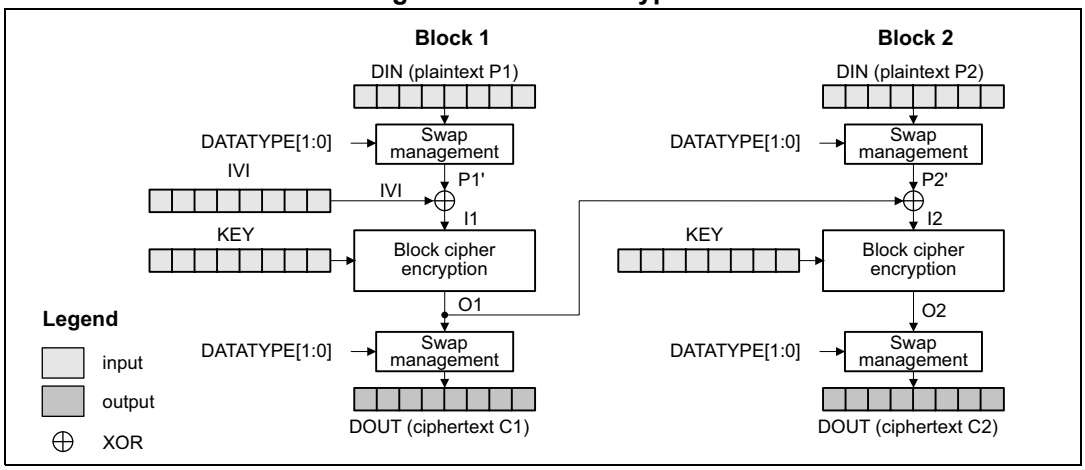
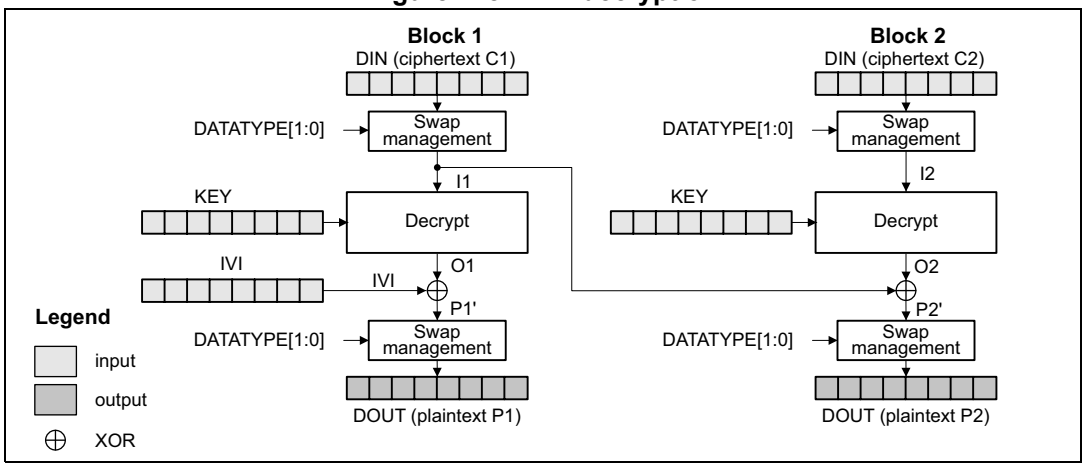


Figure 278. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 32.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 32.4.14* (wrapped keys) and *Section 32.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 32.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 32.4.7*.
   b) Append the cleartext block into SAES as described in *Section 32.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 32.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 32.4.14* (wrapped keys) and *Section 32.4.15* (shared keys).

4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 32.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 32.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 32.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 32.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).

4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

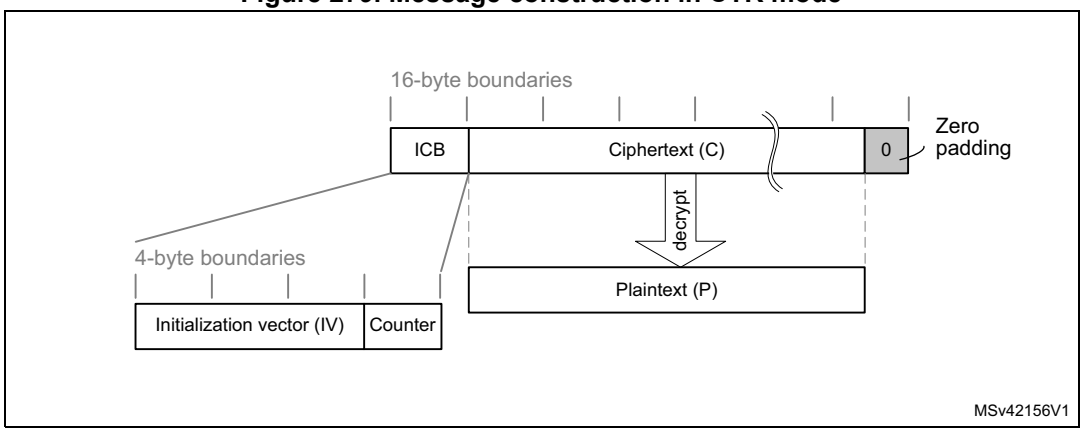
Note: It is not required to save the key registers as the application knows the original key.

32.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 279*.

Figure 279. Message construction in CTR mode



The structure of this message is:

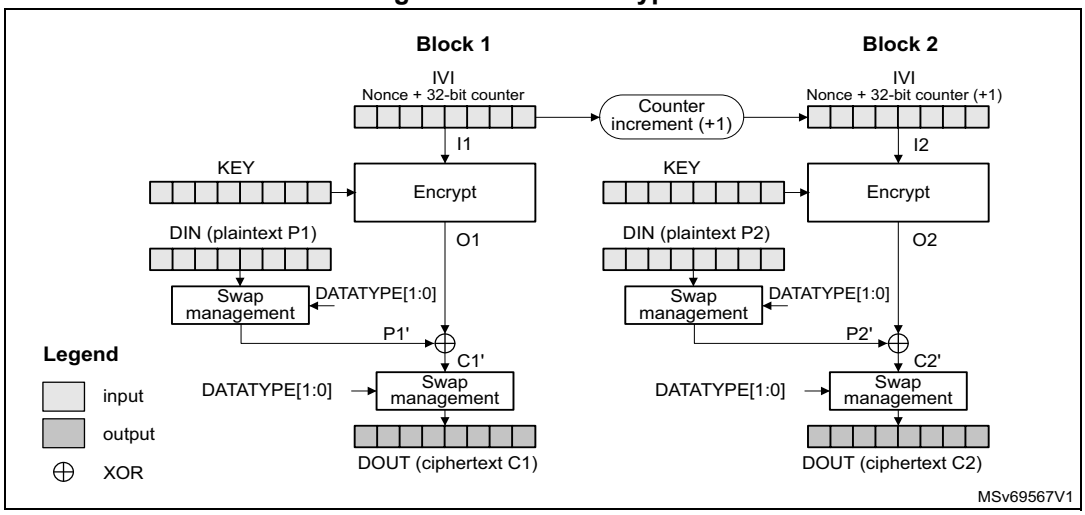
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 280 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 280. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 282*.

Table 282. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 32.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 32.4.14* (wrapped keys) and *Section 32.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 282*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 32.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.

8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 32.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 32.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

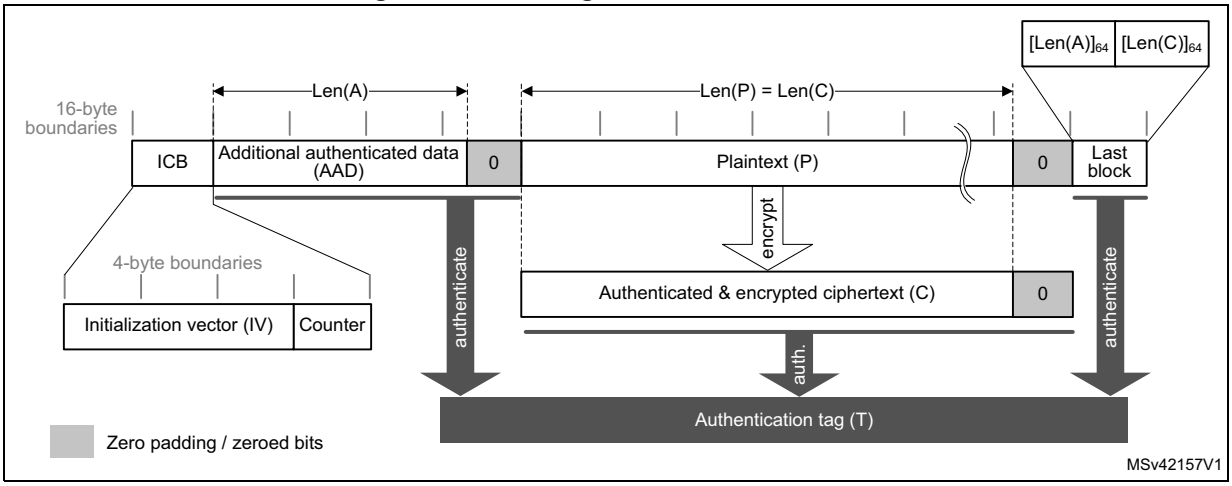
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

32.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 281. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 284*.

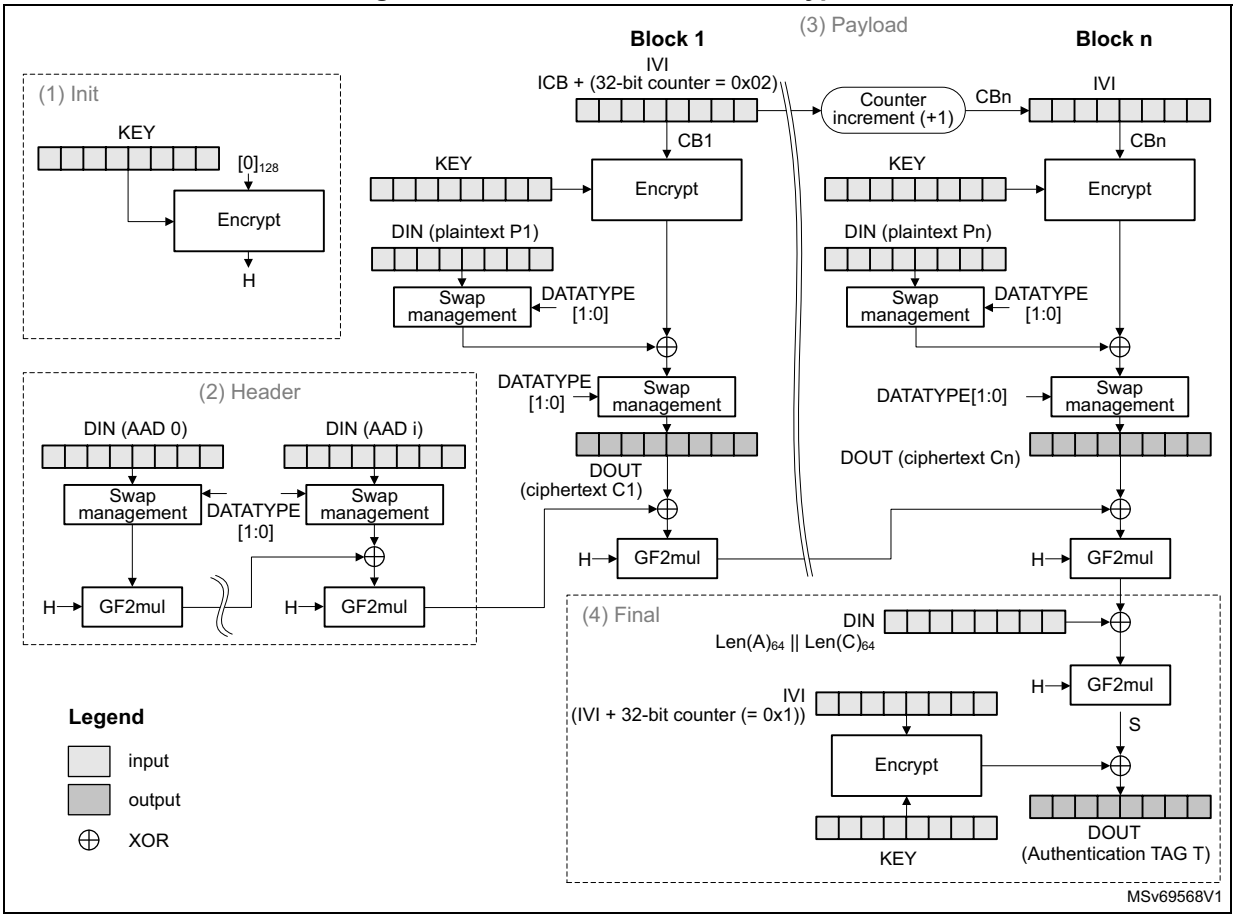
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 282 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 282. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 283*.

Table 283. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 284*.

Table 284. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 32.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, nonsecure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 32.4.14* (wrapped keys) and *Section 32.4.15* (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 283*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 32.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 32.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 32.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 284*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

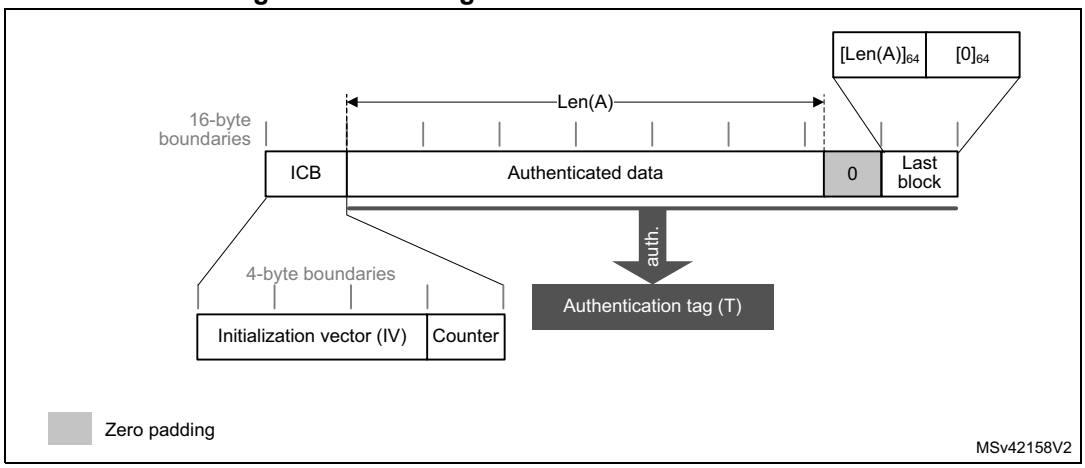
Suspend/resume operations are not supported in GCM mode.

32.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

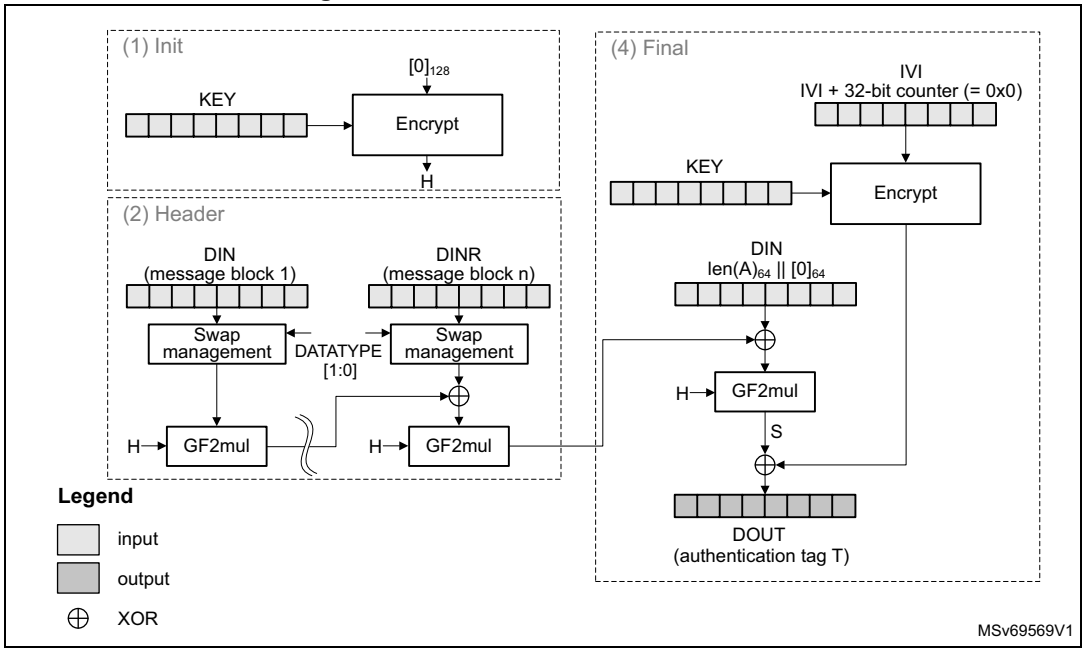
Figure 283. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 284 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 284. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

Suspend/resume operations are not supported in GMAC mode.

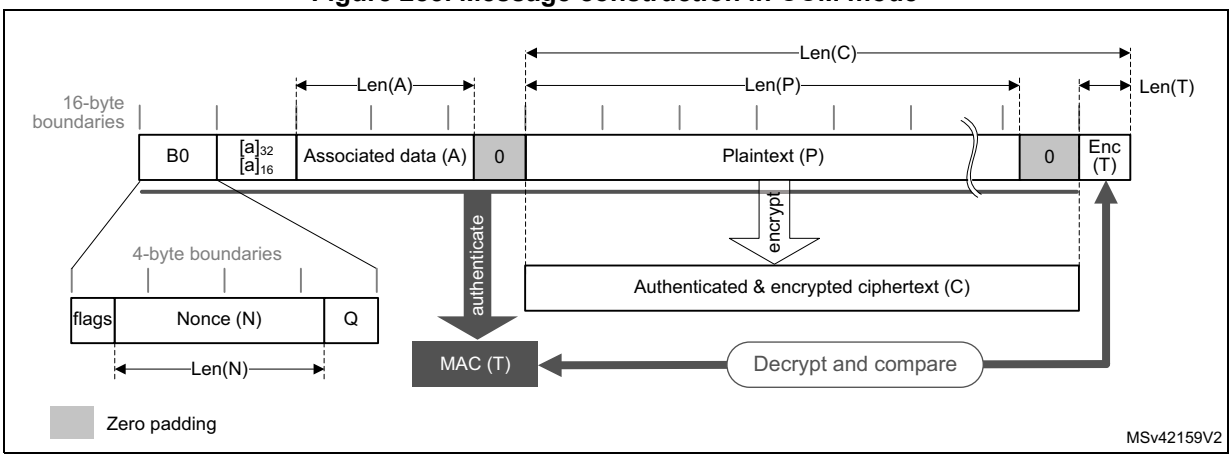
32.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 285. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 285*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 285*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

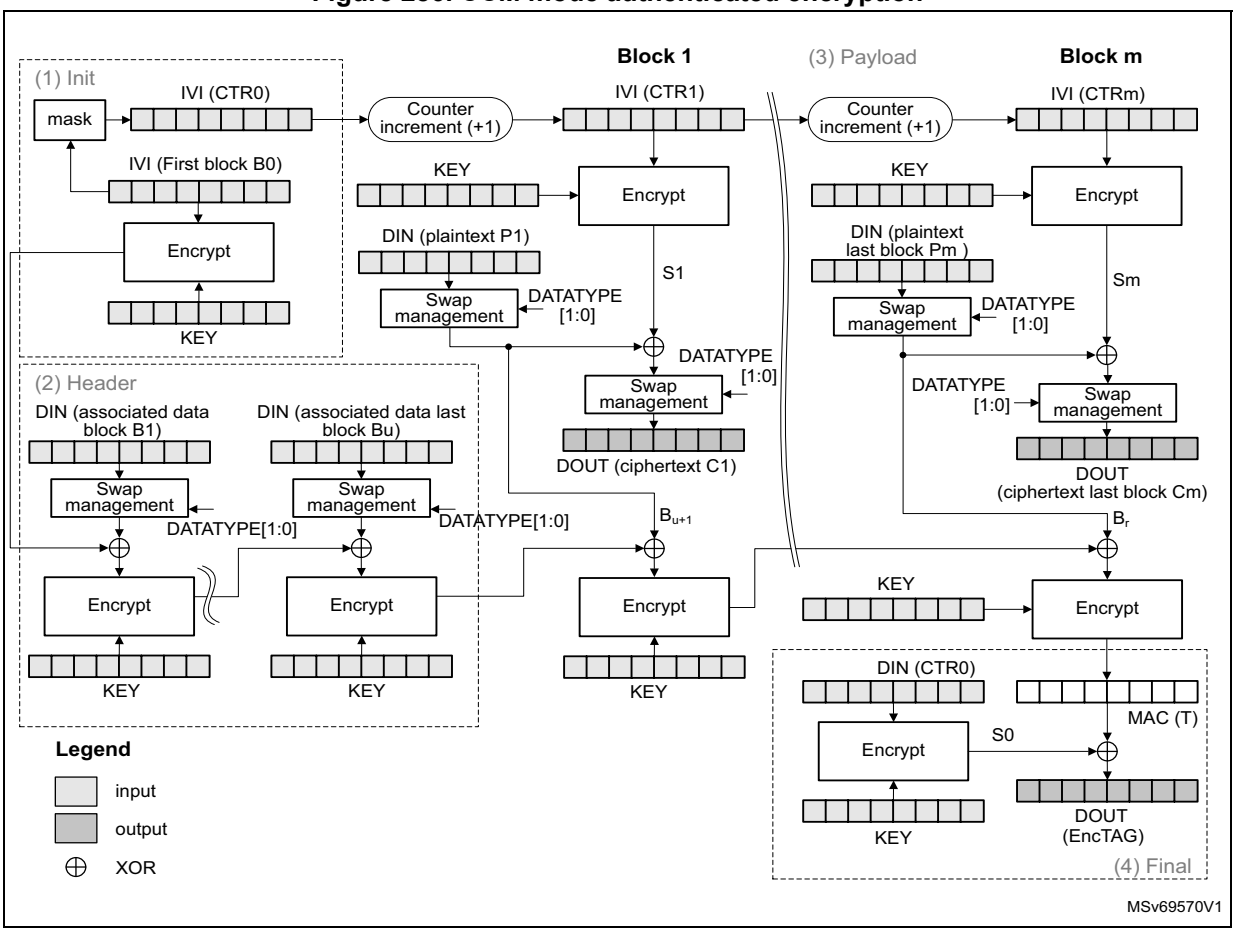
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 286 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 286. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 285*.

Table 285. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 32.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 32.4.14* (wrapped keys) and *Section 32.4.15* (shared keys).
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 285*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 32.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 32.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 32.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 32.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

32.4.14 SAES operation with wrapped keys

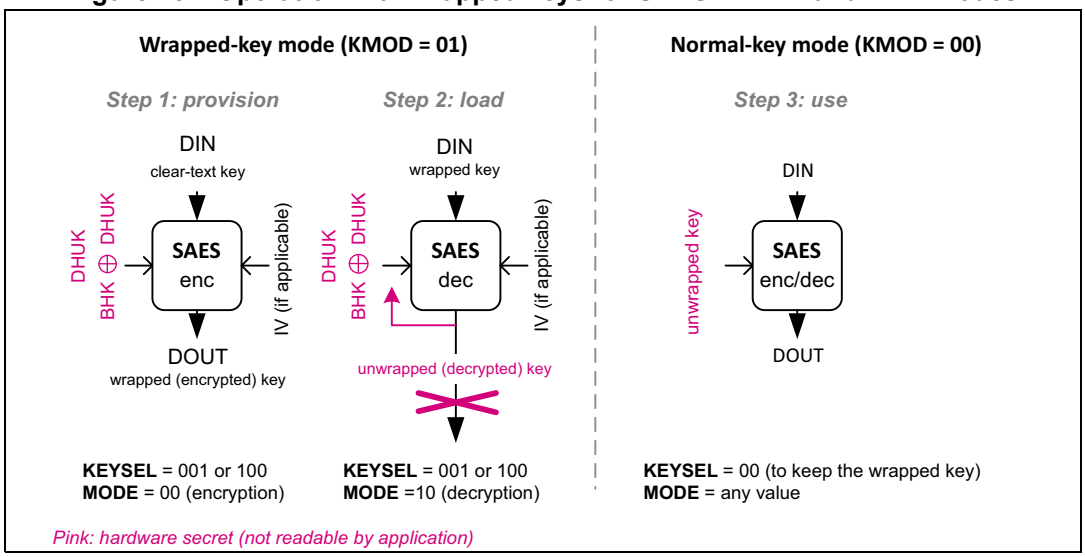
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 287 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 287. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 32.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 287*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 287*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Decrypted wrapped key can be shared with an application running in a different security context (different security attribute) if KEYPROT bit was cleared during step 3.

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

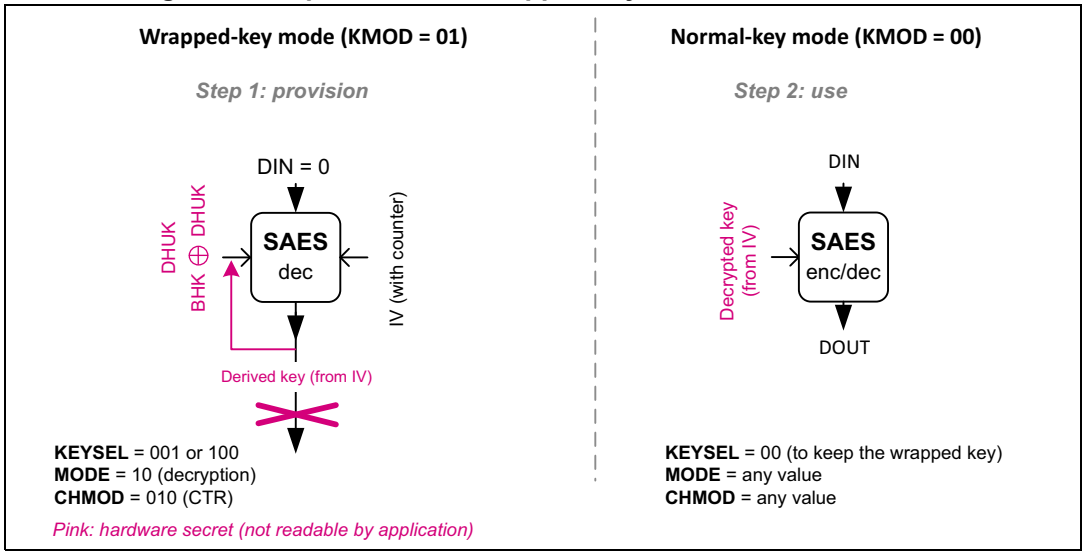
When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 288 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by

writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 288. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 32.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_*CR register.*

32.4.15 SAES operation with shared keys

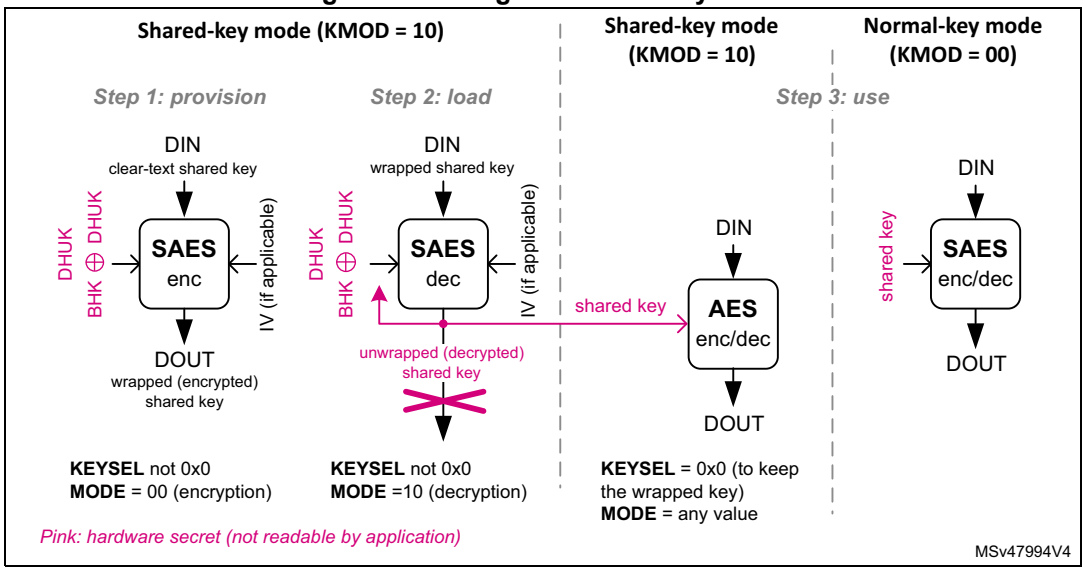
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Note: When a key stored in SAES is shared with AES, the protection given by KEYPROT bit is lost. The protection is detailed in Section 32.4.17: SAES key registers.

Figure 289 summarizes how to wrap or unwrap keys to share with *AES* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 289. Usage of Shared-key mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the AES peripheral.

Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped

key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 289*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 289*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the AES peripheral, and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

32.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 286* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

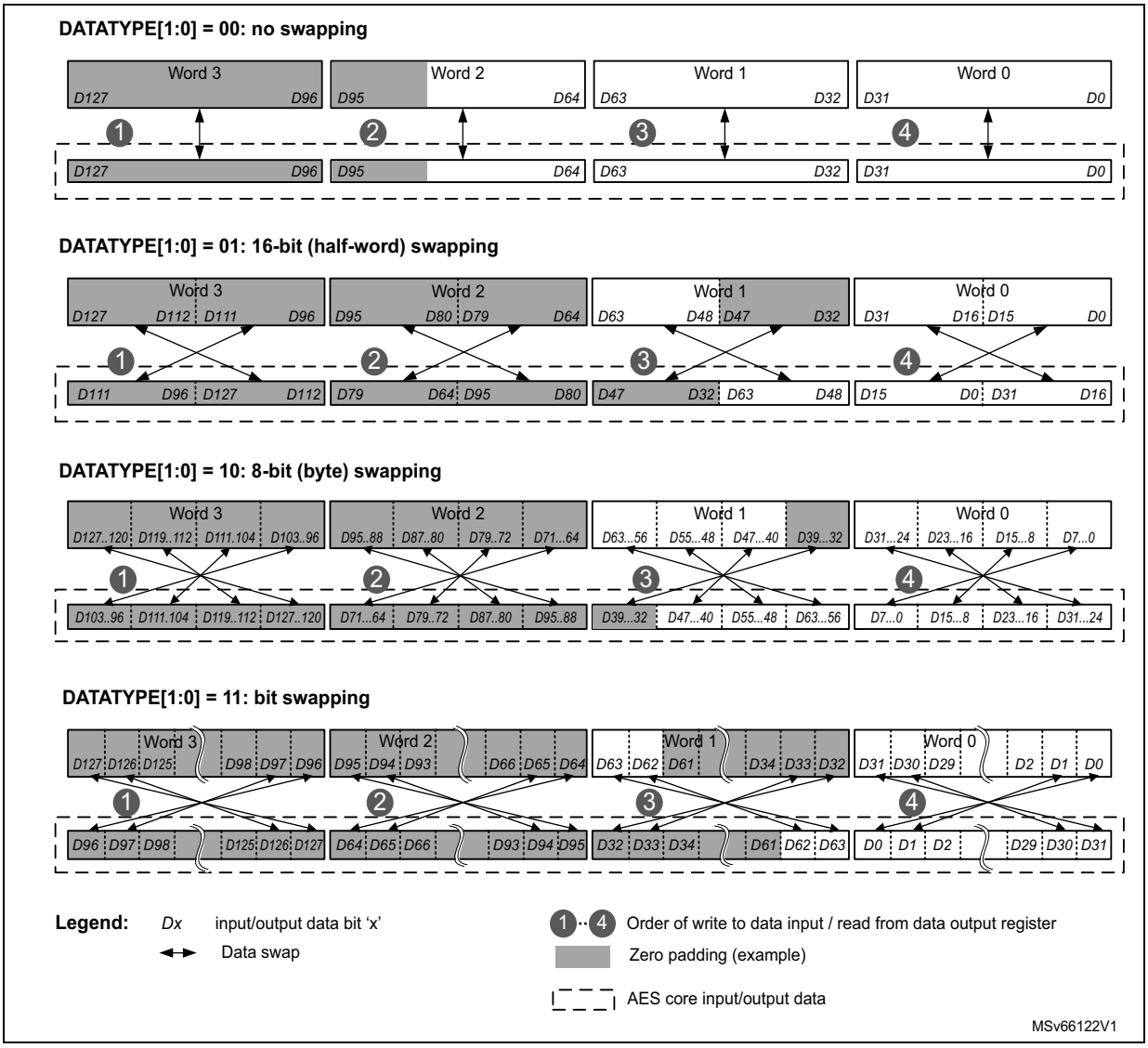
Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 286* and *Figure 290*.

Table 286. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 290. 128-bit block construction according to the data type



Data padding

Figure 290 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

32.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 287*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 287. Key endianness in SAES_KEYRx registers (128/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 32.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a non-volatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, the computed DHUK depends on the SAES usage (privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], CPU, and KEYSIZE context).

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 287*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR

registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 287*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 32.4.19*).

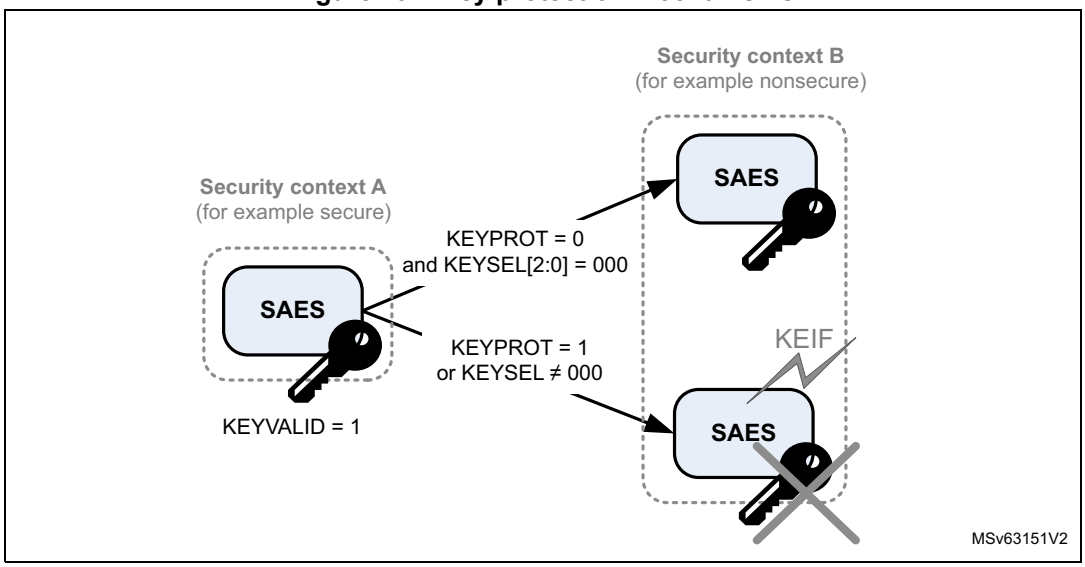
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 291*, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 291. Key protection mechanisms



32.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 288*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 288. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

32.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 32.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 32.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 32.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 32.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 32.5: SAES interrupts*.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 32.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

Managing tamper errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

32.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 32.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 289 gives a summary of the available features.

Table 289. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

32.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 32.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 32.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

32.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 290. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 291. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

32.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

32.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, nonsecure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security context (such as secure or nonsecure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0] = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as nonsecure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, nonsecure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 32.4.16: SAES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

32.8.2 SAES status register (SAES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 32.4.17: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

32.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

32.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

32.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

32.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register

32.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

32.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

32.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

32.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

32.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 32.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

32.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIE:** RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

32.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

32.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag clear

Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the SAES_ISR register.

32.8.21 SAES register map

Table 292. SAES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	SAES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 292. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SAES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	SAES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	SAES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 Hash processor (HASH)

33.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

33.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO, corresponding to a 64-byte block size (16 x 32 bits)
- Automatic padding to complete the input bit string to fit digest minimum block size
- AHB slave peripheral, accessible by 32-bit words only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

33.3 HASH implementation

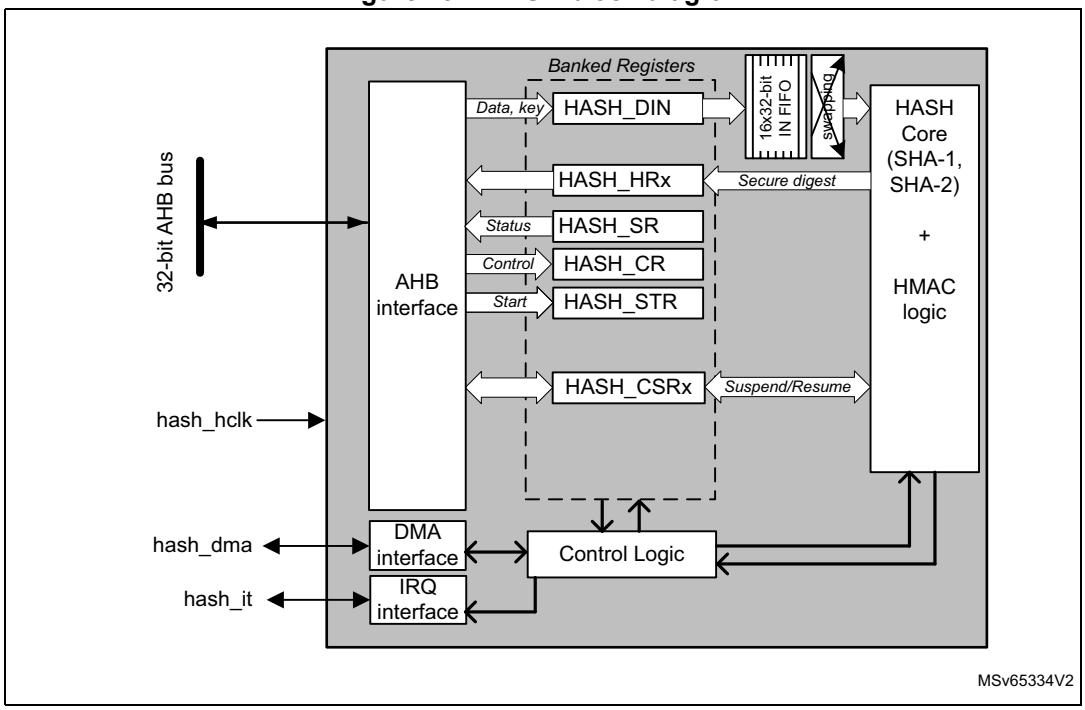
The devices have a single instance of HASH peripheral.

33.4 HASH functional description

33.4.1 HASH block diagram

Figure 292 shows the block diagram of the hash processor.

Figure 292. HASH block diagram



33.4.2 HASH internal signals

Table 293 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 293. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

33.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 294*).

Table 294. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

33.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

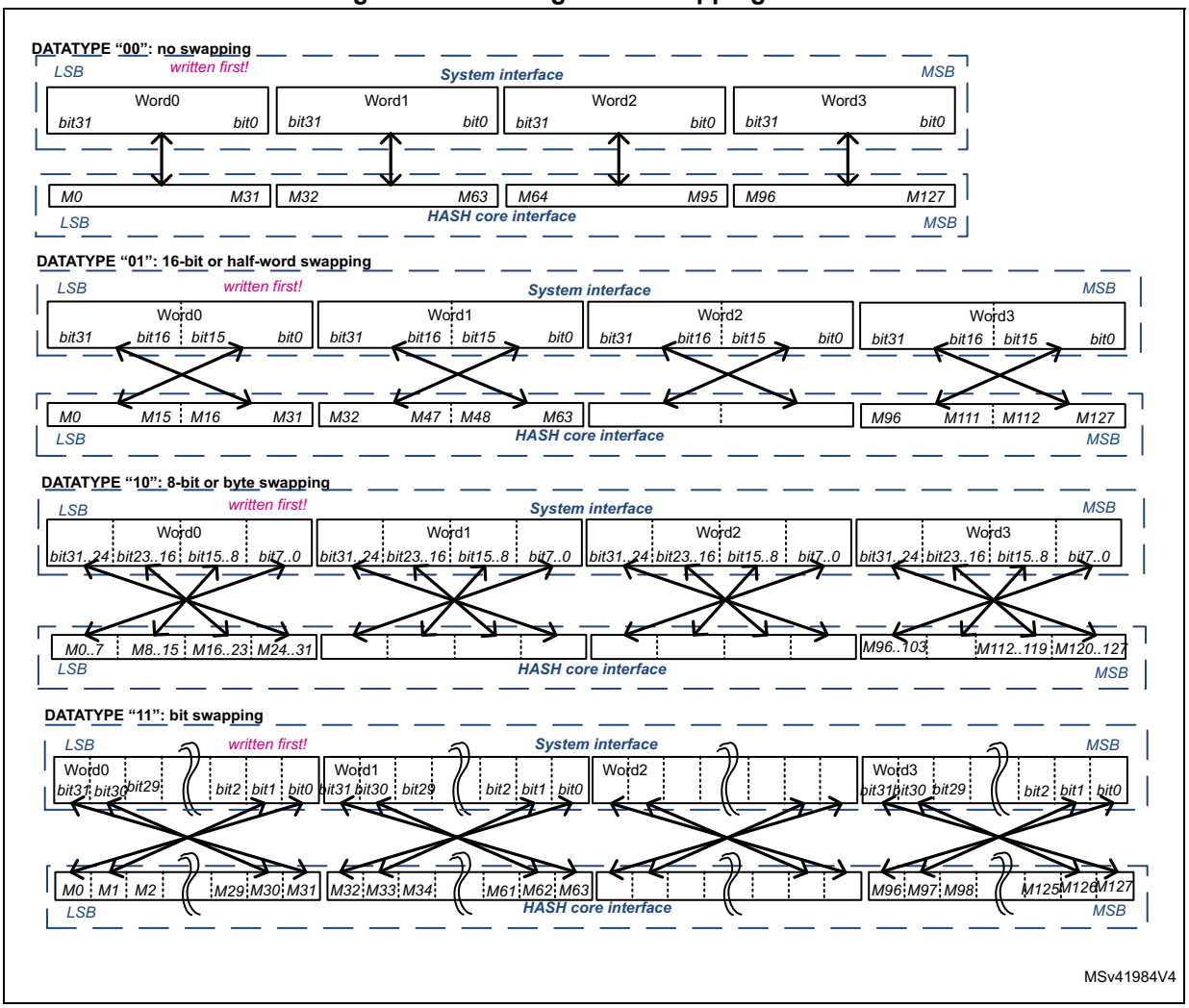
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 293* shows how the hash processing core 32-bit data block M0...31 is

constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 293. Message data swapping feature



33.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 294: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 33.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the

input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **(NBWE-1) * 4**. This step is not required if the block size is already known (see *Table 294* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 33.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 33.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 33.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to

receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 295*. When set, the DMAEN bit is automatically cleared by hardware.

Table 295. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(1)

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 33.4.7: HMAC operation*.

33.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 33.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a "**1**" is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

01100001 01100010 01100011 *1*UUUUUUU

Since L = 24, the number of bits in the above bit string is 25, and 423 "0" bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

61626*38*0 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where '**U**' means don't care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D

33.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

HMAC(message)= Hash((Key | pad) XOR opad |
Hash((Key | pad) XOR ipad | message))

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 33.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 33.4.11: HASH processing time.

Endianness management details can be found in Section 33.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

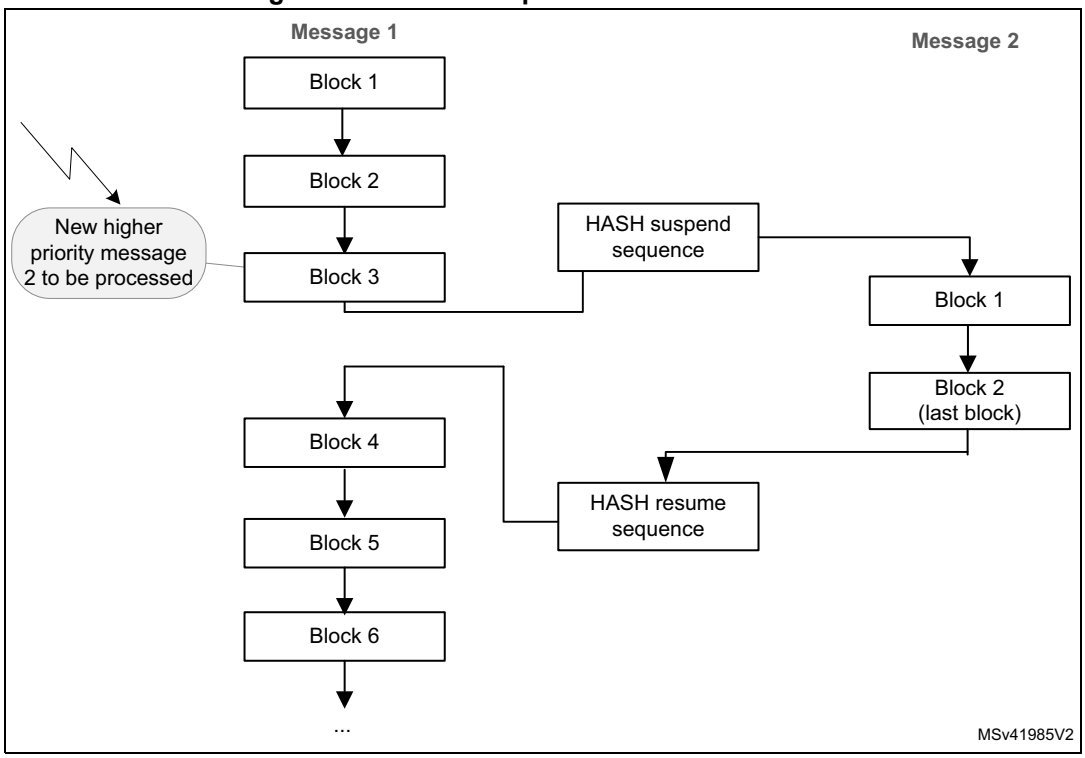
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

33.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 294*.

Figure 294. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 294*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

33.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 33.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

33.4.10 HASH error management

No error flags are generated by the hash processor.

33.4.11 HASH processing time

Table 296 summarizes the time required to process an intermediate block for each mode of operation.

Table 296. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

33.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 297* gives a summary of the available features.

Table 297. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

33.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

33.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	rw		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 294: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bit is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

Setting this bit resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
Clearing this bit has no effect. Reading this bit always returns 0.

Bits 1:0 Reserved, must be kept at reset value.

33.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 294: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

33.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Reading this bit returns zero.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

33.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 33.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 33.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 33.6.4: HASH digest registers* introduction.

33.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

33.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

33.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0020 0000 (HASH_CSR2)

Reset value: 0x0000 0000 (others)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 33.6.7: HASH context swap registers* introduction.

33.6.8 HASH register map

Table 298. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]				DINNE			NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	

Table 298. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 4) Last address: 0x320	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x310 + 0x4 * x, (x = 5 to 15) Last address: 0x3D4	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

34 Public key accelerator (PKA)

34.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, all needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

34.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Support for CCB chaining operations required to protect private key used in PKA protected operations
- Hardware protections to monitor usage of private keys during protected operation initialization

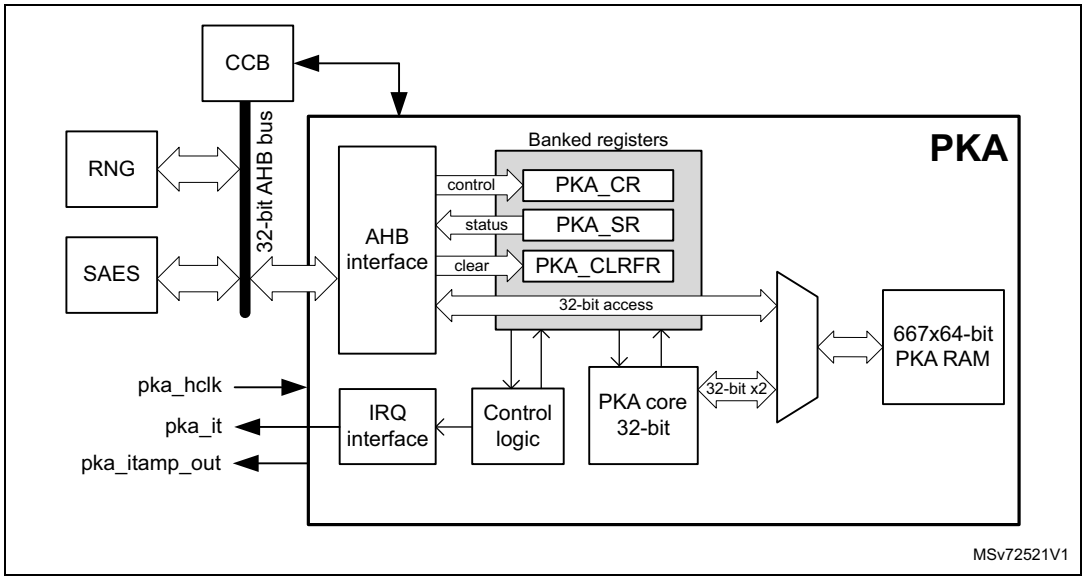
34.3 PKA implementation

The devices have a single instance of the peripheral.
The devices STM32U375xx use the PKA in limited mode (only ECDSA signature verification is available).

34.4 PKA functional description

34.4.1 PKA block diagram

Figure 295. PKA block diagram



34.4.2 PKA internal signals

Table 299 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 299. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). A tamper is also generated if application does not respect the maximum supported size of operands, or when PKA finds a fault in critical configuration registers(1). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

1. When tamper is active the AHB interface is blocked: RAM and register reads are 0-ed and writes are ignored. There is no PKA RAM erase.

34.4.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 2000 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 34.4.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

34.4.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 34.6.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 300* and *Table 301*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 300. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 34.5.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 34.5.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 34.5.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 34.5.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 34.5.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 34.5.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 34.5.7*
0x0D	001101	Modular reduction A mod n	*Section 34.5.8*
0x09	001001	Arithmetic addition A+B	*Section 34.5.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 34.5.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 34.5.11*

Table 300. PKA integer arithmetic functions list (continued)

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 34.5.12*
0x07	000111	RSA CRT exponentiation	*Section 34.5.13*

Table 301. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 34.5.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 34.5.15*
0x23	100011	ECC complete addition	*Section 34.5.18*
0x24	100100	ECDSA sign (protected)	*Section 34.5.16*
0x26	100110	ECDSA verification	*Section 34.5.17*
0x27	100111	ECC double base ladder	*Section 34.5.19*
0x2F	101111	ECC projective to affine	*Section 34.5.20*

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 300* and *Table 301* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in *Section 34.6.2: Computation times*.

The only operation using fast mode is modular exponentiation (MODE = 0x02).

34.4.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For the following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on the above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) * (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process protected version of modular exponentiation (MODE = 0x3) is strongly recommended for security reason. For encryption process MODE = 0x3 cannot be used, as it requires the knowledge of the private key.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 34.6.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, *n* x G = identity element O).

ECDSA message signature generation

ECDSA (elliptic curve digital signature algorithm) signature generation function principle is the following: Alice, to sign a message *m* using her private key integer d_A, goes through the following steps.

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* A^{-1} mod *n*
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A.

Bob can verify that Q_A is a valid curve point going through the following steps:

1. check that Q_A is not equal to the identity element O
2. check that Q_A is on the agreed curve
3. check that n x Q_A = O.

Then Bob follows the procedure detailed below:

1. verify that *r* and *s* are integer in [1, n-1]
2. calculate *e* = HASH(*m*), where HASH is the agreed cryptographic hash function
3. let z be the L_n leftmost bits of *e*
4. calculate w = s^{-1} mod *n*
5. calculate u_1 = zw mod *n* and u_2 = *rw* mod *n*
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if *r* = x_1 (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

34.4.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Note: When PKA is in the process of clearing its memory EN bit cannot be set.

When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 34.4.8).

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 34.5*.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error, and step *3* (PROCENDF = 1) never happens. See *Section 34.4.8* for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in *Section 34.4.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 34.5.5*). This is know as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 34.6.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 34.5*.

34.4.7 Private keys protection using CCB

Chaining modes

As shown in *Figure 296*, when the application sets CCOP to a value different from 0 in the CCB peripheral, a chaining mechanism is automatically setup between PKA, RNG or SAES, depending on the OPSTEP value (see *Table 302*). When the chaining feature is activated, CCEN bit is set in PKA_SR register.

Methods to go through the various OPSTEP for each protected operations (ECDSA signature, scalar multiplication, modular exponentiation) are defined in the CCB coupling and chaining operations of the CCB section in this document.

Figure 296. Chaining of PKA with SAES or RNG

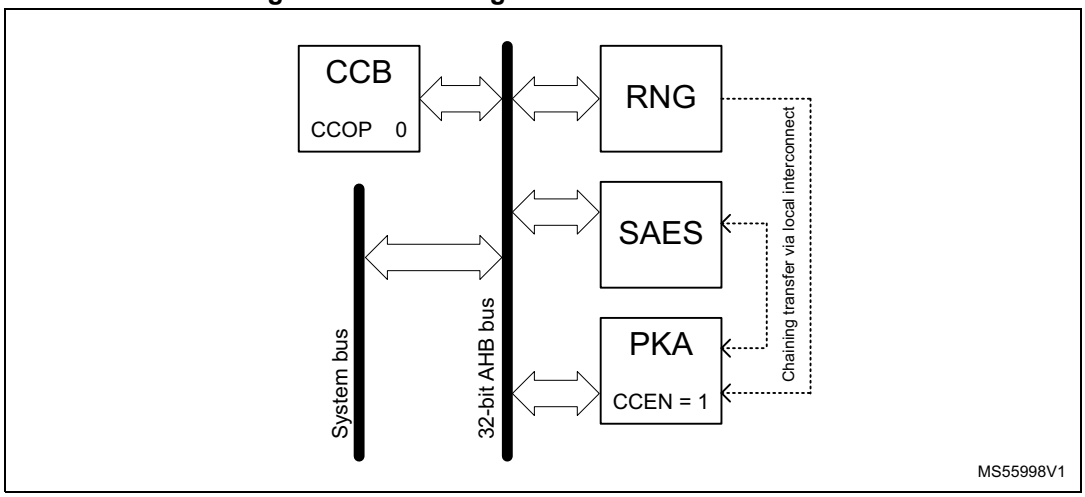


Table 302. Chaining mode to/from PKA

CCOP		OP STEP	Expected MODE(1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
x	0xC3	0x1	0x24	N/A	Hash (unprotected)	N/A
ECDSA signature	0xC0 or 0xC2	0x3	0x24	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0xC3	0x13				
		0x14		SAES → PKA	Decrypted private key d	DATAOKF or CMF error
		0x16		RNG → PKA	Random integer k	RNGOKF or CMF error
	0xC0	0x4		N/A	Clear text private key d	DATAOKF or CMF error
	0xC2	0x6		RNG → PKA	Random private key d	RNGOKF or CMF error
	0xC0 or 0xC2	0x8		PKA → SAES	Clear-text private key d	DATAOKF or CMF error
Scalar multiplication	0x80 or 0x82	0x3	0x20	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x81	0x13				
		0x14		SAES → PKA	Decrypted k	DATAOKF or CMF error
	0x80	0x04		N/A	Clear text scalar mult k	
	0x82	0x6		RNG → PKA	Random scalar mult k	RNGOKF or CMF error
	0x80 or 0x82	0x8		PKA → SAES	Clear-text scalar mult k	DATAOKF or CMF error

Table 302. Chaining mode to/from PKA (continued)

CCOP		OP STEP	Expected MODE(1)	Chaining traffic(2)	Protected parameter written to PKA RAM	Expected flags(3)
Modular exponentiation	0x44	0x3	0x3	CPU → PKA + SAES(4)	Integrity protected parameters(5)	CMF error only
	0x45	0x13				
		0x14		SAES → PKA	Decrypted exponent e	DATAOKF or CMF error
		0x15			Decrypted phi	
	0x44	0x4		N/A	Clear text exponent e	
		0x5			Clear text phi	
		0x8		PKA → SAES	Clear text exponent e	
		0x9			Clear text phi	

1. This mode must be written in PKA_CR before the START bit is set in PKA_CR.
2. Chaining traffic triggered by the CPU access to PKA RAM. “N/A” means no chaining traffic, only CPU traffic.
3. In PKA_SR bits 8 to 15 (“CMF error” refers to error flag bits 8 to 14).
4. In this mode chaining traffic includes the 32-bit data word plus the 32-bit physical address to PKA RAM.
5. Defined in *Table 303*.

When the application uses a protected operation in PKA (with its private key), it must load the related blob information in the PKA RAM and in the SAES_DINR register. If the information exactly matches the conditions of the blob generation (choice and order of the writes of protected parameters in PKA RAM, selection of blob encryption key, usage of the original encrypted secrets) the CCB allows it to set the START bit in PKA_CR. When PKA ignores the setting of START bit, the application must reset the CCB using IPRST bit in CCB_CR, then start the protected operation sequence again.

General protected application usage is described hereafter. It is composed of three phases.

Step 1: Public phase

Before using a protected operation, the application may need to write unprotected parameters to the PKA RAM, as defined in *Table 302*.

Step 2: Integrity protected phase

To use a PKA operation, the application must load specific parameters in PKA RAM. When CCEN bit is set in PKA_SR, the integrity protected parameters defined in *Table 303* must always be written in the same way. Integrity protection is enforced in specific OPSTEPs, as defined in *Table 302*.

Typical issues with integrity of protected operation parameters are:

- The blob encryption key loaded in SAES registers is not correct.
- At least one confidentiality protected parameter defined in *Table 304* has an incorrect value, or was not properly written (DATAOKF = 0 or RNGOKF = 0, see *PKA status register (PKA_SR)*).
- At least one integrity protected parameter defined in *Table 303* has an incorrect value, has been written at an incorrect address, or its write sequence to PKA RAM does not match the sequence used during blob creation. On this specific point the application

must follow the sequence detailed in *Table 303* when writing each item described in the column *Parameters*.

CCB feeds integrity protected parameters to SAES during the header phase of the AES-GCM (GCMPH = 0x1).

Table 303. Integrity protected parameters list

Protected operation blob[1]	Parameters
ECC key blob	Curve prime order n length, curve modulus p length, curve coefficient a sign, curve coefficient |a|, curve coefficient b, curve modulus value p, curve prime order n
ECC signature blob	Same as scalar multiplication operation, adding curve base point G coordinates x and y
RSA key blob	Exponent e length, operand length, modulus value n

1. See *Table 304* for more details.

Step 3: Confidentiality phase

To use any PKA protected operation, the application must load confidentiality protected parameters in PKA RAM. When CCEN bit is set in PKA_SR the parameters defined in *Table 304* are protected in confidentiality by:

- making PKA RAM write-only
- triggering a MODE error (MDERRF) in PKA_SR when MODE bit in PKA_CR does not match the expected value when application sets START bit in PKA_CR
- monitoring the correct management of protected parameters through DATAOKF and RNGOKF flags (see *PKA status register (PKA_SR)*)
- optionally, generating ECC secrets from the on-chip TRNG when CCOP = 0xC2 or 0x82 and OPSTEP = 0x6, and when CCOP = 0xC3 and OPSTEP = 0x16. PKA performs a sanity check before accepting this random value (see RNGERRF bitfield).

Confidentiality protected parameters are described in *Table 304*. CCB feeds them to SAES during the payload phase of the AES-GCM (GCMPH = 0x2).

Table 304. Confidentiality protected parameter list

PKA operation	Secrets	Protected operation blob[1]	Start offset in PKA RAM	Size[2]
Modular exponentiation (protected, MODE = 0x3)	Exponent e	RSA key blob	@0x14B8	ROS
	Phi value		@0xC68	
Scalar multiplication (MODE = 0x20)	Scalar k	ECC key blob	@0x12A0	EOS
ECDSA signature (MODE = 0x24)	Integer k	Not available	@0x12A0	EOS
	Private key d	ECC signature blob	@0xF28	

1. Information blob where confidentiality and integrity protected parameters are stored (see also *Table 303*)
2. See *Section 34.5.1* for details

Step 4: Running PKA protected operation

Last step is to set START bit in PKA_CR. If overall sequence has been properly executed PKA RAM becomes not readable nor writable. Input parameters are also automatically erased when PROCENDF flag is set in PKA_SR, or application aborts the current operation using the EN bit in PKA_CR.

34.4.8 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the address error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM error flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return 0, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (see bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.
- When CCEN bit is set, the running chaining mode sequence triggered at least one error. When this error occurs, CMF (chaining mode flag) is set in the PKA_SR register, along with one of the following status bits (see bitfield description in *Section 34.8.2*):
  - Increment error (INCRERRF)
  - Data 0-ed error (DATAZF)
  - Trailing zeros error (TRZERRF)
  - Mode error (MDERRF)
  - RNG error (RNGERRF)

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 34.7* for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

When CMF flag is set, with one of the corresponding error status bit set, the application must restart the running protected operation sequence in the CCB, after setting the CMFC bit in PKA_CLRFR register.

Note: The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

34.5 PKA operating modes

34.5.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.

Warning: The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified in the operation tables.

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size* / 64 + 1) words, with *rsa_size* equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand size): data size is (*ecc_size* / 64 + 1) words, with *ecc_size* equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

Note: The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation *ECC Fp scalar multiplication*, when the application writes the x coordinate of point P for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information a = -3, on a curve with a modulus length of 224 bits (that is, four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

```
@RAM+410   0x0000000000000001  /* curve coefficient 'a' sign without extra word */
@RAM+418   0x0000000000000011  /* value of |a| LSB
@RAM+420   0x0000000000000000  ...
@RAM+428   0x0000000000000000  ...
```

@RAM+430 0x0000000000000000 value of |a| MSB */

@RAM+438 0x0000000000000000 /* additional all 0 word */

34.5.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod *n*) used by PKA to convert operands into the Montgomery residue system representation. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in *Table 305*.

Table 305. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod n	-	RAM@0x620	

34.5.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 306*.

Table 306. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xE78	

34.5.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 307*.

Table 307. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(n < 2^{4160})	RAM@0x1088	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xE78	

34.5.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Assuming that A is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – Result AR= A x *r2modn* mod *n* is A in the Montgomery domain.
   b) Assuming that BR is an integer in the Montgomery domain:
      – Result B = BR x 1 mod *n* is B in the natural domain.
      – Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n.*
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute AB= AR x B mod *n*. Output is in natural domain.
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain.
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain.
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain.
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain.
   g) (optional) Repeat the two steps above if more operands need to be multiplied.
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain.

Operation instructions for Montgomery multiplication are summarized in *Table 308*.

Table 308. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
OUT	Result: AxB mod n(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

34.5.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod *n*.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 309* and *Table 310*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 311*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 309* (normal mode), *Table 310* (fast mode) and in *Table 311* (protected mode). Fast mode usage is explained in *Section 34.4.6*.

Table 309. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 310. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
IN/OUT	Montgomery parameter R2 mod n	(mandatory)	RAM@0x620	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 311. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x03	PKA_CR	6 bits
	Exponent e length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0x16C8	ROS
	Exponent e	(0 ≤ e < n)	RAM@0x14B8	
	Modulus value n	(odd integer only, n < 2^{4096})	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	
ERROR	Error A^e mod n	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n^1 with phi = (p - 1) * (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (*p*, *q*, *e*, *d*) instead of (*N*, *e*, *d*, *phi*), in this case, to derive *N* and *phi* using PKA arithmetic multiplier: *N* = *p* * *q*, and *phi* = (*p* - 1) * (*q* - 1).

34.5.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse A^{-1} mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the greatest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 312*.

Table 312. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xE78	

34.5.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 313*.

Table 313. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand A	(0 ≤ A < 2n < 2^{4160})	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result A mod n	(0 < result < n)	RAM@0xE78	

34.5.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 314*.

Table 314. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < 2^M)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < 2^M)	RAM@0xC68	
OUT	Result: A+B	(0 ≤ result < 2^{M+1})	RAM@0xE78	ROS + 1

34.5.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If A ≥ B result equals A - B
- If A < B and M/32 residue is > 0 result equals A + $2^{int(M/32)*32+1}$ - B
- If A < B and M/32 residue is 0 result equals A + $2^{int(M/32)*32}$ - B

For the last two bullets the 32-bit word following the most significant word of the output equals 0xFFFF FFFF, as result is negative.

Operation instructions are summarized in *Table 315*.

Table 315. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xC68	
OUT	Result: A-B	($0 \leq$ result $< 2^M$)	RAM@0xE78	

34.5.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 316*.

Table 316. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < 2^M$)	RAM@0xC68	
OUT	Result: AxB	($0 \leq$ result $< 2^{2M}$)	RAM@0xE78	2xROS

34.5.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in *Table 317*.

Table 317. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < 2^M$)	RAM@0xC68	
OUT	Result A?B	0xED2C, 0x7AF8 or 0x916A	RAM@0xE78	64 bits

34.5.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- d_P= d mod (*p* -1)
- d_Q= d mod (*q* -1) and
- q_{inv}= q^{-1} mod *p*

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- m_1 = A^{dP} mod *p*
- m_2 = A^{dQ} mod *q*
- h = q_{inv} (m_1 - m_2) mod *p*, with m_1> m_2
- m = m_2 + hq mod *pq*

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 318*.

Table 318. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	(0 < d_P < $2^{M/2}$)	RAM@0x730	ROS / 2
	Operand d_Q	(0 < d_Q < $2^{M/2}$)	RAM@0xE78	
	Operand q_{inv}	(0 < q_{inv} < $2^{M/2}$)	RAM@0x948	
	Prime *p*(1)	(0 < p < $2^{M/2}$)	RAM@0xB60	
	Prime *q*(1)	(0 < q< $2^{M/2}$)	RAM@0x1088	
IN	Operand A	(0 < A< 2^M)	RAM@0x12A0	ROS
OUT	Result: A^d mod *pq*	(0 ≤ result < *pq*)	RAM@0x838	

1. Must be different from 2.

34.5.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation y^2 = (x^3 + *ax* + *b*) mod *p,* where *a* and *b* are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 319*.

Table 319. Point on elliptic curve Fp check

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < p < 2640)	RAM@0x470	
	Point P coordinate *x*	(x < *p*)	RAM@0x578	
	Point P coordinate *y*	(y < *p*)	RAM@0x5D0	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Result: point P on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: x or y coordinate is not smaller than modulus p	RAM@0x680	64 bits

34.5.15 ECC Fp scalar multiplication

This operation consists in the computation of k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 320*.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 320. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 320. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier k	($0 \leq k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x < p$)	RAM@0x578	
	Point P coordinate y_P	($y < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: k x P coordinate *x'*	(result < *p*)	RAM@0x578	
	Result: k x P coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error k x P	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For k = 0 this function returns a point at infinity (0, 0) if curve parameter b is nonzero, (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation must be carried out.
- For k < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-k| must be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

34.5.16 ECDSA sign

ECDSA signing operation (outlined in *Section 34.4.5*) is summarized in *Table 321* (input parameters) and in *Table 322* (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new k must be generated and the ECDSA sign operation must be repeated.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 321. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient b	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer k(1)	(0 ≤ k < n)	RAM@0x12A0	
	Curve base point G coordinate x	(x < *p*)	RAM@0x578	
	Curve base point G coordinate y	(y < *p*)	RAM@0x470	
	Hash of message z	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key d	(0 < d < n)	RAM@0xF28	
	Curve prime order n	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases k can be deterministically generated.
2. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 322. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x730	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (*Table 321* and *Table 322*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 323*.

Table 323. Extended ECDSA sign - Extra outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

34.5.17 ECDSA verification

ECDSA verification operation (outlined in *Section 34.4.5*) is summarized in *Table 324* (input parameters) and *Table 325* (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

Table 324. ECDSA verification - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0< p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 325. ECDSA verification - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– (0 < *r* < n)	RAM@0x578	EOS

34.5.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in *Table 326*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y)= (X/Z^2,Y/Z^3) can be used.

Table 326. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate X	(x < *p*)	RAM@0xD60	
	Result coordinate Y	(y < *p*)	RAM@0xDB8	
	Result coordinate Z	(z < *p*)	RAM@0xE10	

34.5.19 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized in *Table 327*.

If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 327. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer k	($0 < k < 2^{640}$)	RAM@0x520	
	Integer m	($0 < m < 2^{640}$)	RAM@0x578	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate x	(x < *p*)	RAM@0x578	EOS
	Result coordinate y	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

34.5.20 ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (X, Y, Z), it computes the affine coordinates (x, y) = (X/Z, Y/Z).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 328*.

Table 328. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate X (projective)	(x < *p*)	RAM@0xD60	
	Point P coordinate Y (projective)	(y < *p*)	RAM@0xDB8	
	Point P coordinate Z (projective)	(z < *p*)	RAM@0xE10	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Point P coordinate x (affine)	(x < *p*)	RAM@0x578	
	Point P coordinate y (affine)	(y < *p*)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

34.6 Example of configurations and processing times

34.6.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b \pmod{p}$.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 329 summarizes the family of curves supported by PKA for ECC operations.

Table 329. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

34.6.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 330. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield= 0x07).

Table 331. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 332. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 333. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 334. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 335. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 336. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	53000	89000

Table 337. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 338. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

34.7 PKA interrupts

There are five individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 34.4.8*
2. Access to unmapped address (ADDRERRF), see *Section 34.4.8*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 34.4.8*
4. PKA end of operation (PROCENDF)
5. Chaining mode flags (CMF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

CMF interrupt is triggered by the end of an operation (DATAOKF, RNGOKF) or by a chaining mode error (see *Section 34.4.8*). Chaining modes are described in *Section 34.4.7*.

Table 339 gives a summary of the available features.

Table 339. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit
	Chaining mode flag	CMF	CMFIE	Set CMFC bit

34.8 PKA registers

34.8.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
									rw	rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFIE**: Chaining mode flags interrupt enable
0: No interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.
1: An interrupt is generated when any flag in bits 8 to 15 is set in PKA_SR.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA. PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

34.8.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
									r	r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF	Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	r	r	r	r	r	r	r						r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMF**: Chaining mode flags

This status bit is only used when CCEN status bit is set in this register.

0: No chaining mode events.

1: Running chaining mode sequence triggered at least one flag status in bits 8 to 15 of this register.

This bit is cleared using CMFC bit in PKA_CLRFR.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

When CCEN = 1 PKA sets BUSY bit when it runs a sanity check on a random number that CCB wrote to the PKA RAM during a chaining sequence (see *Section 34.4.7*).

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RNGERRF**: RNG error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No random number drawn from RNG, or generated ECC private key is NIST compliant.

1: Random number drawn from RNG did not pass NIST “key pair generation by testing candidates” equivalent algorithm (see FIPS 186-4 annex B.4.2). The last chaining operation must be restarted, otherwise the selected PKA operation remains unusable.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 13 **MDERRF**: Mode error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No mode error

1: The value written in MODE of PKA_CR registers is different from the expected value corresponding to the CCOP mode selected in the CCB peripheral. Refer to *Section 34.4.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 12 **TRZERRF**: Trailing 0s error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No trailing 0s error

1: Clear-text coming from CPU, or decrypted data coming from SAES does not have the required trailing 0s.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 11 **DATAZF**: Data 0-ed error flag

This read-only chaining mode error flag is active only when CCEN is set in this register. DATAZF is set when PKA finds the complete data written to PKA RAM equal to 0, with all the expected 32-bit words null. This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 10 **INCRERRF**: Increment error flag

This read-only chaining mode error flag is active only when CCEN is set in this register.

0: No increment error

1: During the reading or writing of protected parameters in PKA RAM the application did not properly incremented the address. Refer to *Section 34.4.7* for details.

During the read or write of an operand, and before DATAOKF or RNGOKF is set, INCRERRF flag stays cleared if the application reads or writes again this operand (starting from the lowest address). This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 9 **DATAOKF**: Data OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Data not OK (if applicable), otherwise no event.

1: Sequential writes to / read from PKA RAM successfully transfered the expected amount of data from / to the SAES. Alternatively CPU wrote the correct amount of data in PKA RAM. Refer to *Section 34.4.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bit 8 **RNGOKF**: RNG OK flag

This read-only chaining mode status flag is active only when CCEN is set in this register.

0: Random number not OK (if applicable). No event otherwise.

1: Sequential writes to PKA RAM successfully transfered the expected amount of random data from the RNG. Refer to *Section 34.4.7* for details.

This bit is cleared when CMFC bit is set in PKA_CLRFR.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CCEN**: Coupling and chaining mode enable

This bit is set when application configures the CCB to run coupling and chaining sequences that performs protected PKA operation using the SAES and (optionally) the RNG.

0: Access to PKA RAM and programming of PKA_CR has no extra limitation.

1: PKA RAM is write-only until START is set (it is no access then), and the MODE value written in PKA_CR must correspond to the CCOP mode selected in the CCB peripheral. Application must also follow required coupling and chaining sequences described in *Section 34.4.7*.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE bitfield can be used.

1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

Note: When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

34.8.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
									w	w	w	w		w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **CMFC**: Clear chaining mode flag

0: No action

1: Clear the CMF flag and the chaining mode status bits in 8 to 15 in PKA_SR

Bit 21 **OPERRFC**: Clear operation error flag

0: No action

1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag

0: No action

1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag

0: No action

1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

34.8.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 34.4.3: PKA reset and clocks.

34.8.5 PKA register map

Table 340. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFIE	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value										0	0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMF	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	RNGERRF	MDERRF	TRZERRF	DATAZF	INCRERRF	DATAOKF	RNGOKF		Res.	Res.	Res.	Res.	Res.	CCEN	LMF	INITOK
	Reset value										0	0	0	0		0	0		0	0	0	0	0	0	0						0	0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMFC	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0		0																	

Refer to *Section 2.3* for the register boundary addresses.

35 Advanced-control timers (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

35.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 35.3.30: Timer synchronization*.

35.2 TIM1 main features

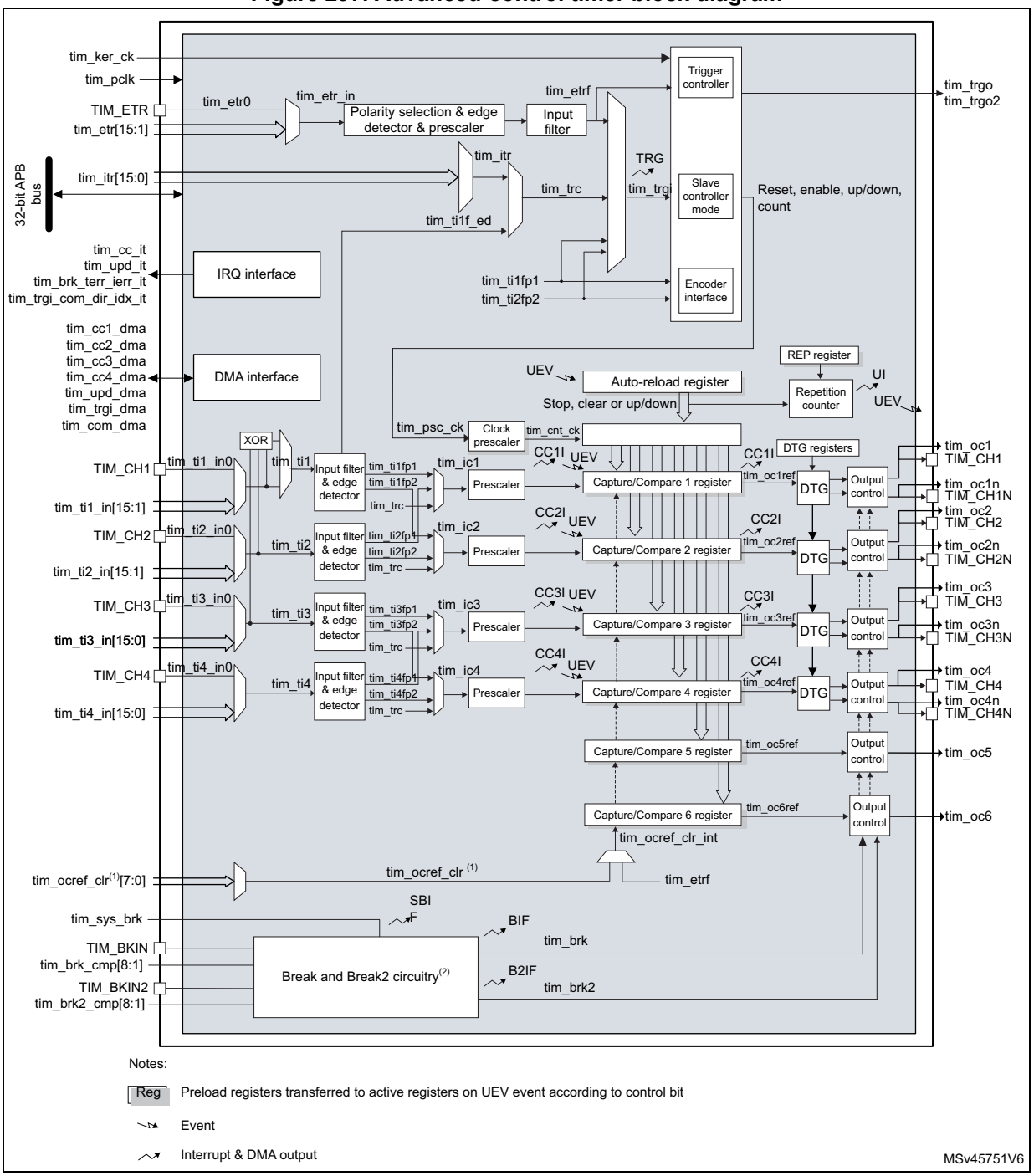
TIM1 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
- ADC synchronization for jitter-free sampling points

35.3 TIM1 functional description

35.3.1 Block diagram

Figure 297. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 35.3.2: TIM1 pins and internal signals*.
2. See *Figure 344: Break and Break2 circuitry overview* for details.

35.3.2 TIM1 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 341. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input/output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 342. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 342. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_brk2_cmp[8:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 343. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources
	TIM1
tim_ti1_in0	TIM1_CH1
tim_ti1_in1	comp1_out
tim_ti1_in2	comp2_out
tim_ti1_in[15:3]	Reserved

Table 344. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM1
tim_ti2_in0	TIM1_CH2
tim_ti2_in[15:1]	Reserved

Table 345. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources
	TIM1
tim_ti3_in0	TIM1_CH3
tim_ti3_in[15:1]	Reserved

Table 346. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources
	TIM1
tim_ti4_in0	TIM1_CH4
tim_ti4_in[15:1]	Reserved

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 347. Internal trigger connection

Timer internal trigger input signal	TIM1
tim_itr0	Reserved
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	Reserved
tim_itr5	Reserved
tim_itr6	tim15_trgo
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr[15:9]	Reserved

Tables below list the internal sources connected to the tim_etr input multiplexer.

Table 348. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment
	TIM1
tim_etr0	TIM1_ETR
tim_etr1	comp1_out
tim_etr2	comp2_out
tim_etr3	MSIK
tim_etr4	HSI
tim_etr5	Reserved
tim_etr6	
tim_etr7	
tim_etr8	adc1_awd1
tim_etr9	adc1_awd2
tim_etr10	adc1_awd3
tim_etr11	adc2_awd1
tim_etr12	adc2_awd2
tim_etr13	adc2_awd3
tim_etr[15:14]	Reserved

Tables below list the sources connected to the tim_brk and tim_brk2inputs.

Table 349. Timer break interconnect

tim_brk inputs	TIM1
TIM_BKIN	TIM1_BKIN pin
tim_brk_cmp1	comp1_out
tim_brk_cmp2	comp2_out
tim_brk_cmp[8:3]	Reserved

Table 350. Timer break2 interconnect

tim_brk2 inputs	TIM1
TIM_BKIN2	TIM1_BKIN2 pin
tim_brk2_cmp1	comp1_out
tim_brk2_cmp2	comp2_out
tim_brk2_cmp[8:3]	Reserved

Table 351. System break interconnect

tim_sys_brk inputs	TIM1	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex-M33 LOCKUP	CLL
tim_sys_brk1	Programmable voltage detector (PVD)	PVDL
tim_sys_brk2	SRAM parity error	SEL
tim_sys_brk3	Flash memory double ECC error	ECCL
tim_sys_brk4	Clock security system (CSS)	None (always enabled)

The table below lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 352. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signal assignment
	TIM1
tim_ocref_clr0	comp1_out
tim_ocref_clr1	comp2_out
tim_ocref_clr[7:2]	Reserved

35.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 298 and *Figure 299* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 298. Counter timing diagram with prescaler division change from 1 to 2

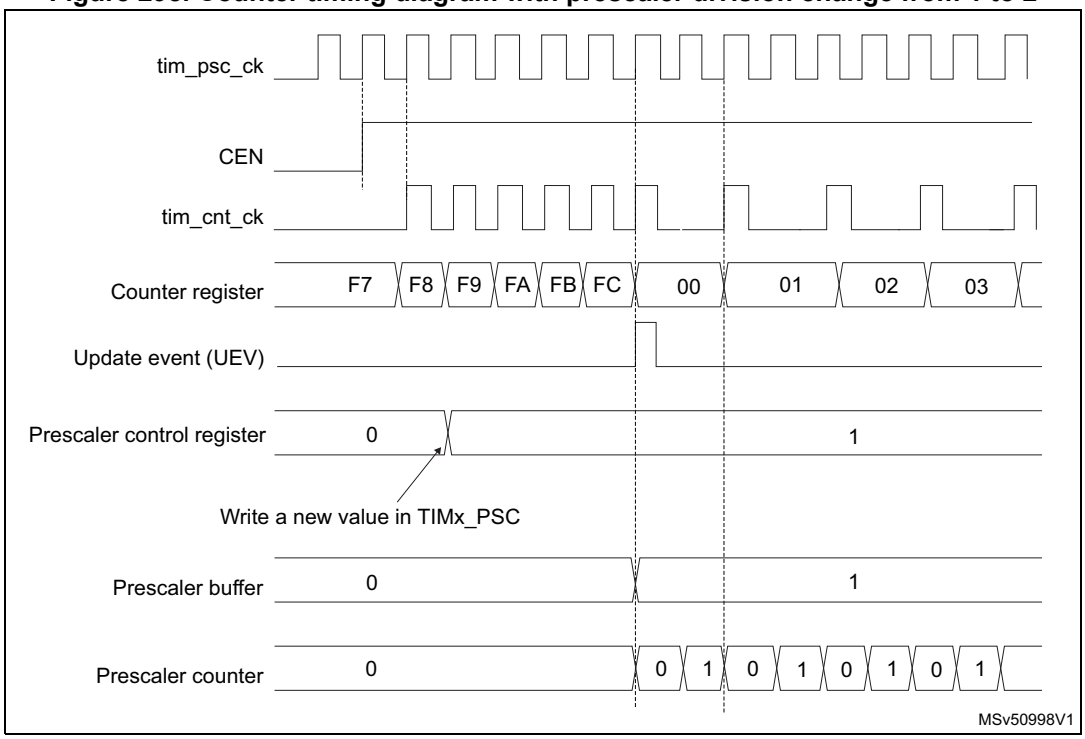
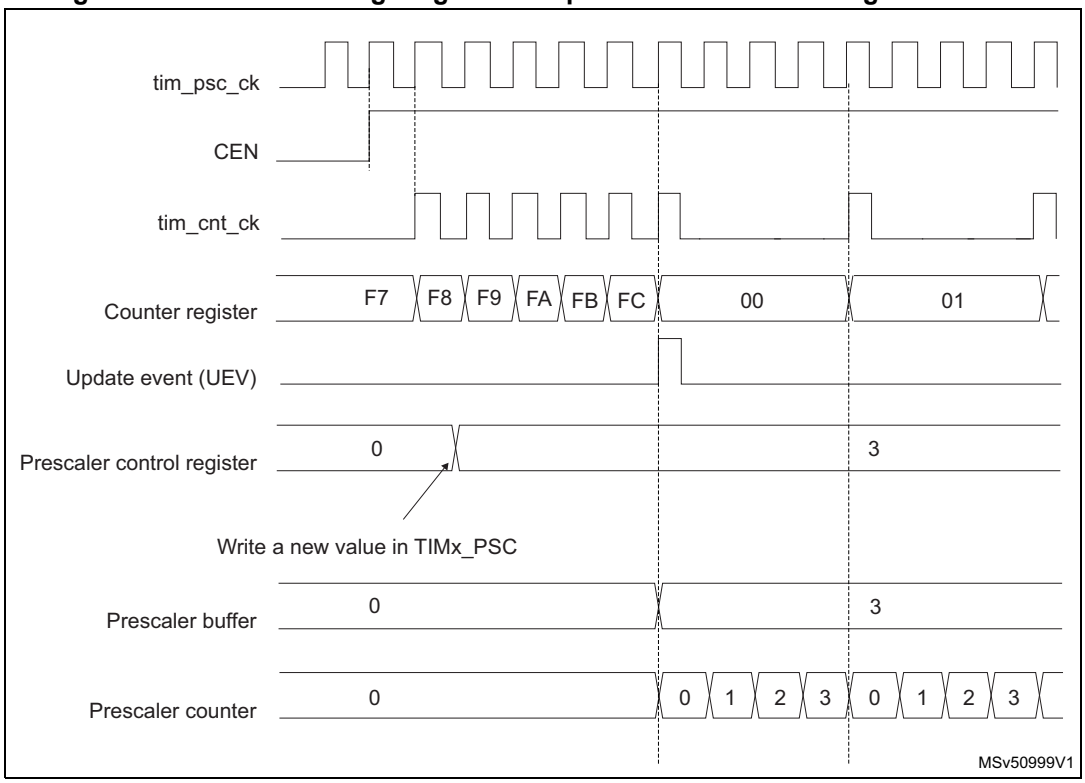


Figure 299. Counter timing diagram with prescaler division change from 1 to 4



35.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 300. Counter timing diagram, internal clock divided by 1

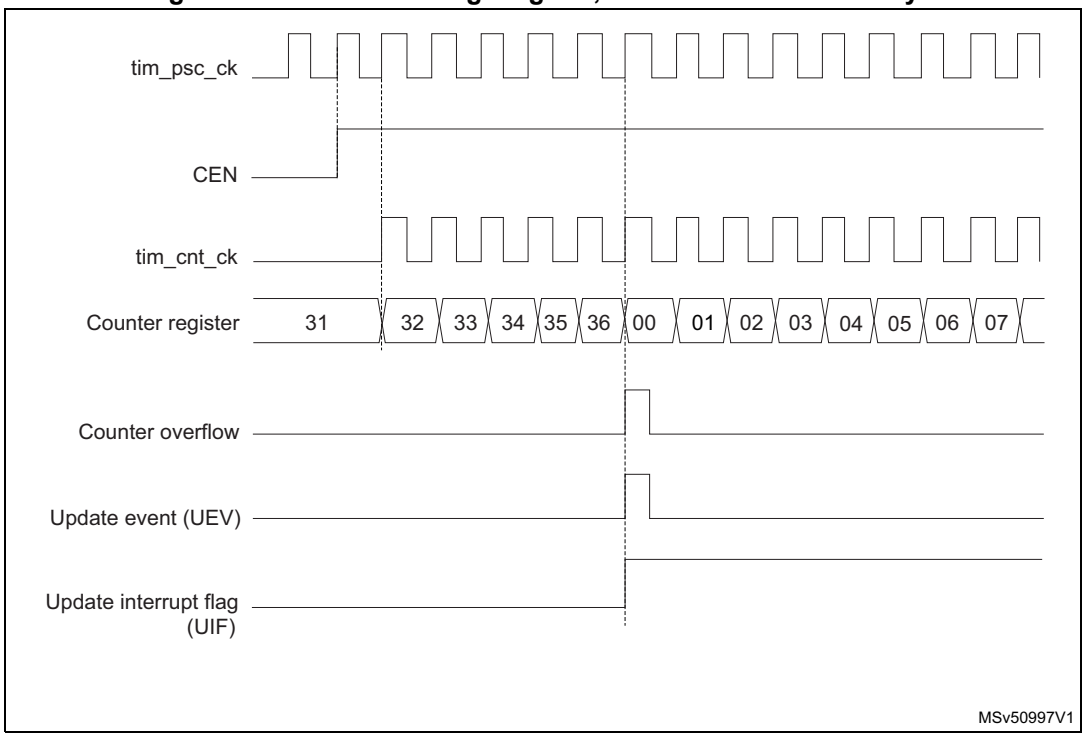


Figure 301. Counter timing diagram, internal clock divided by 2

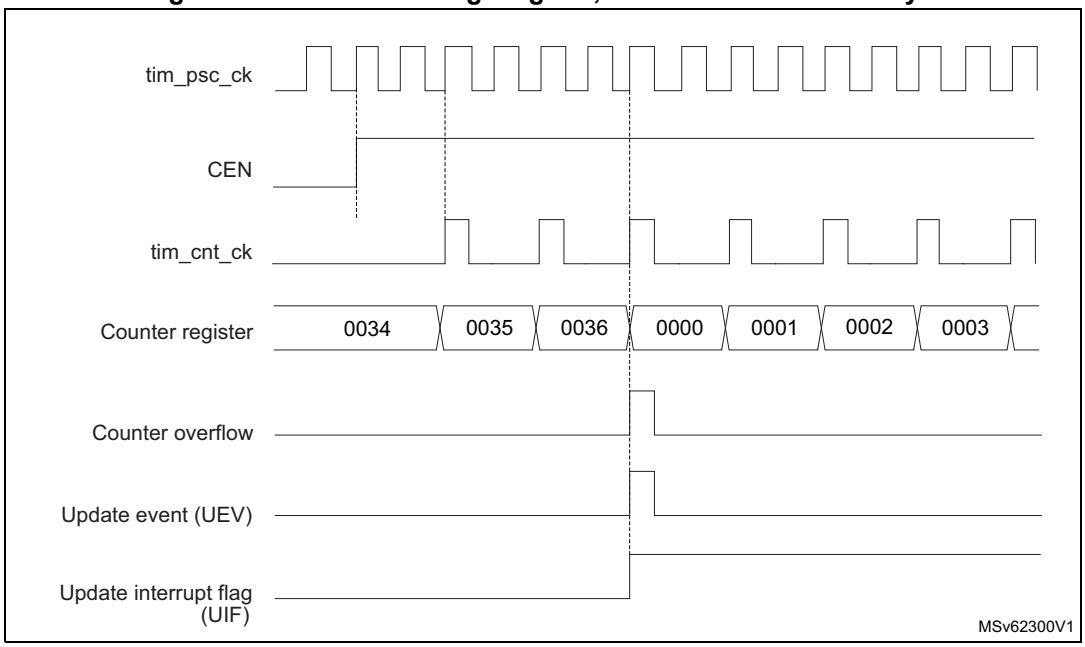


Figure 302. Counter timing diagram, internal clock divided by 4

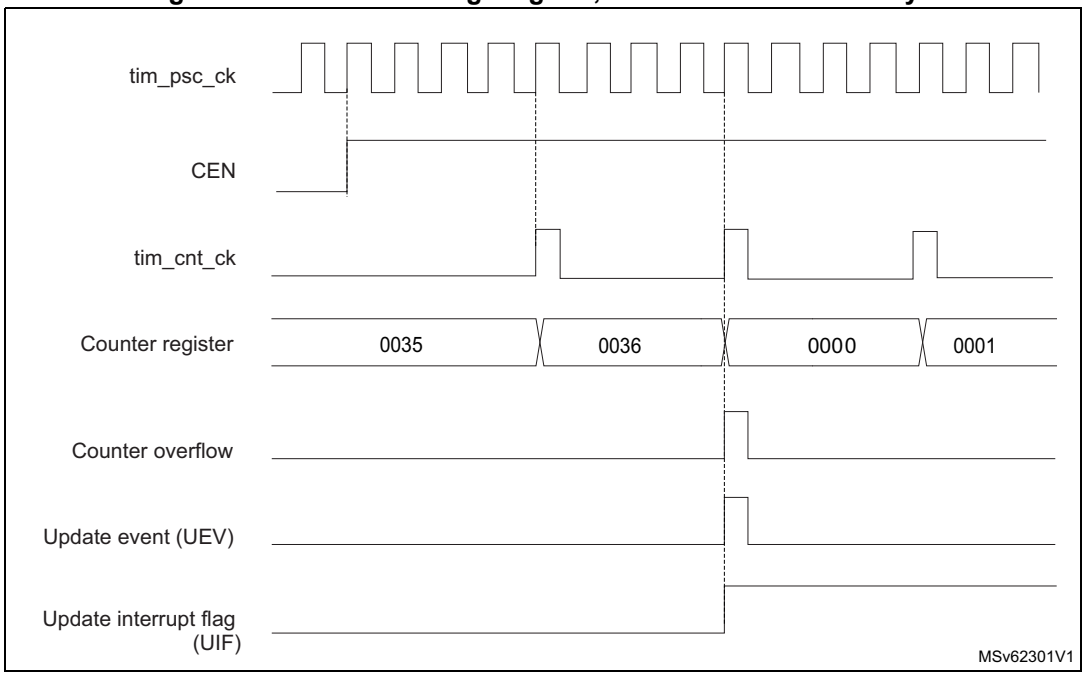


Figure 303. Counter timing diagram, internal clock divided by N

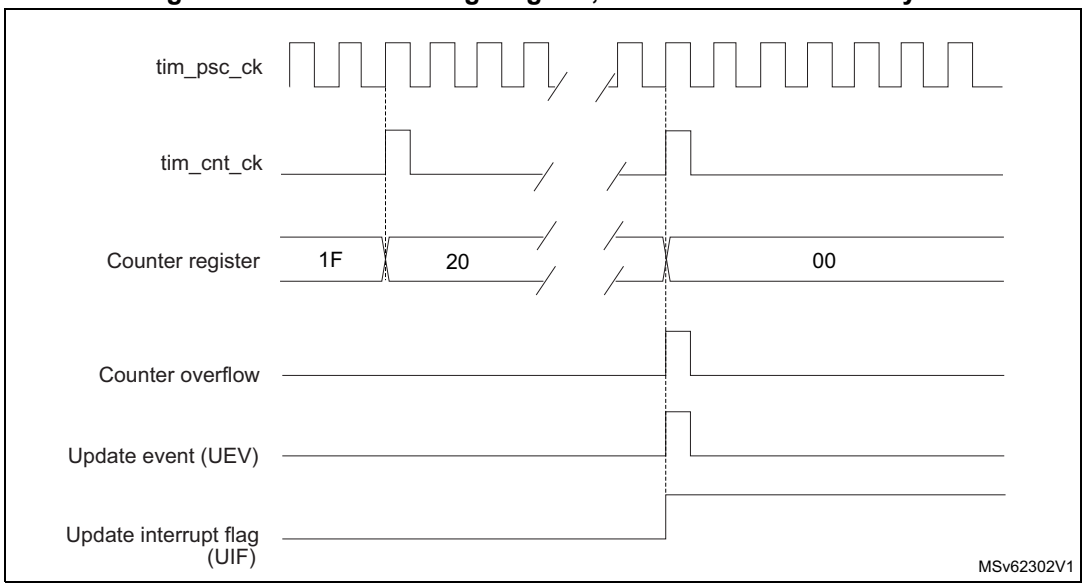


Figure 304. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

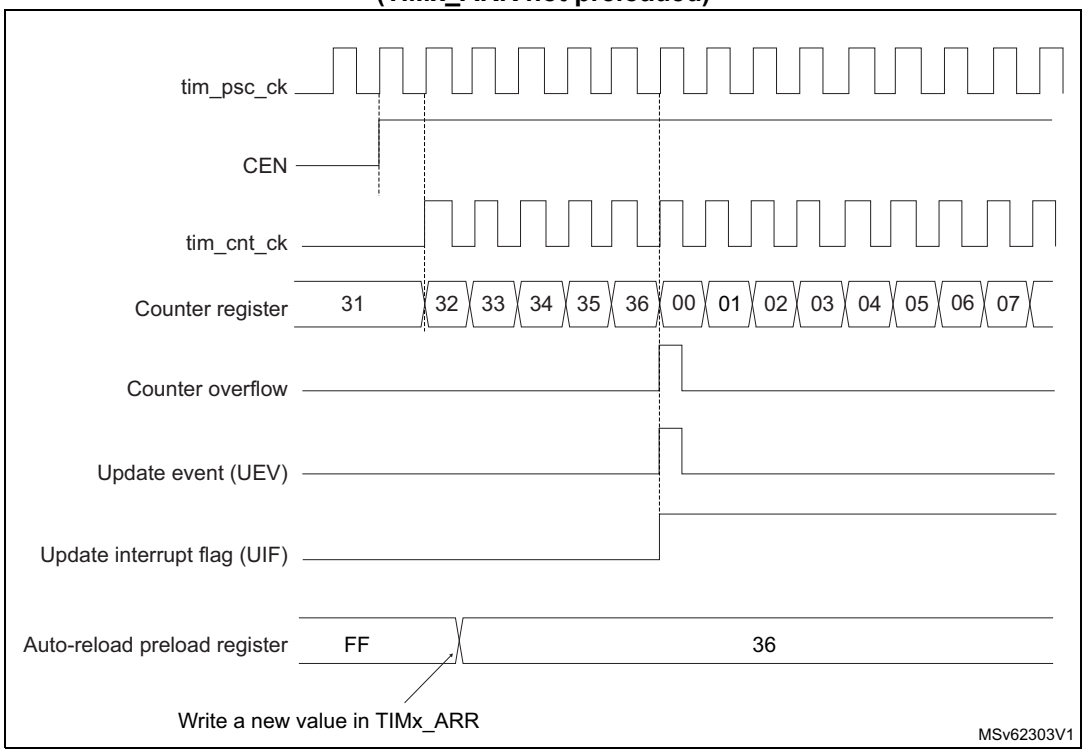
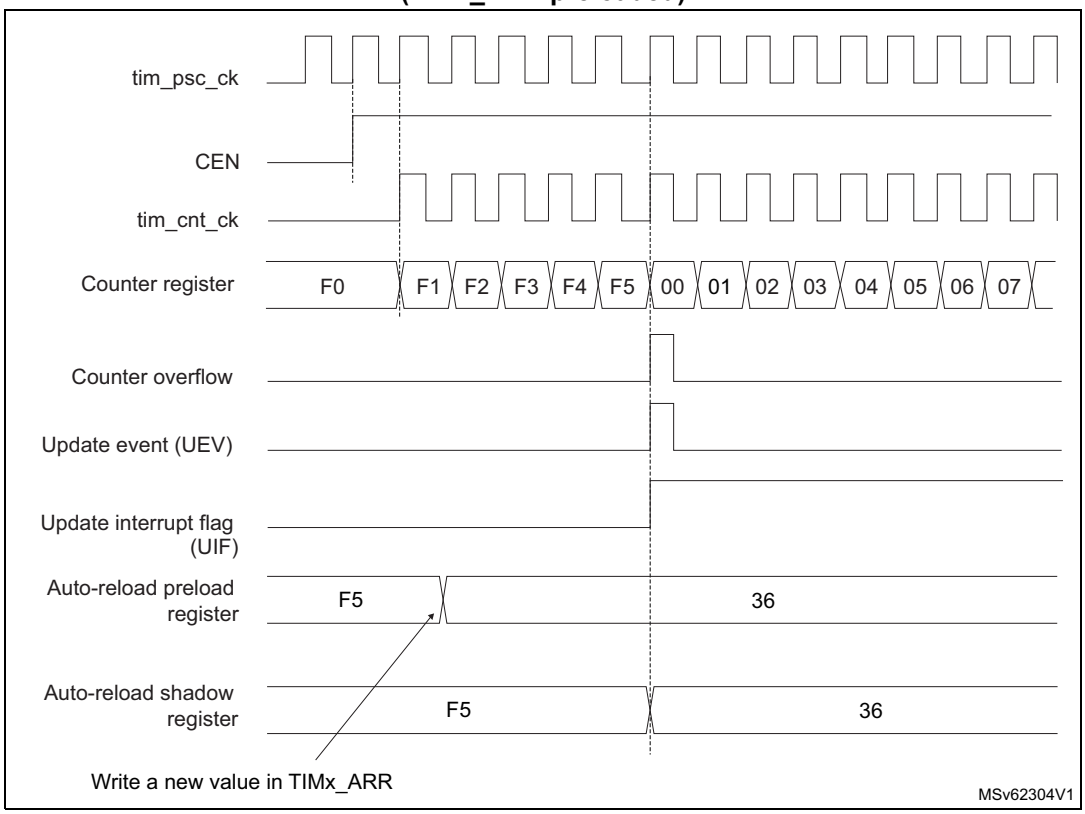


Figure 305. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 306. Counter timing diagram, internal clock divided by 1

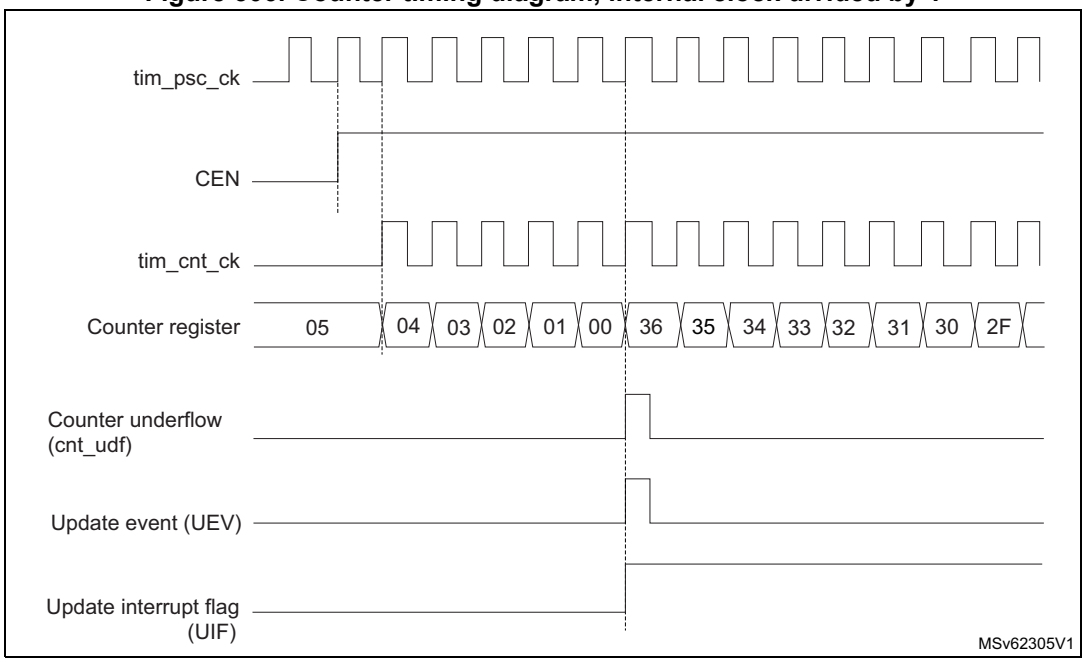


Figure 307. Counter timing diagram, internal clock divided by 2

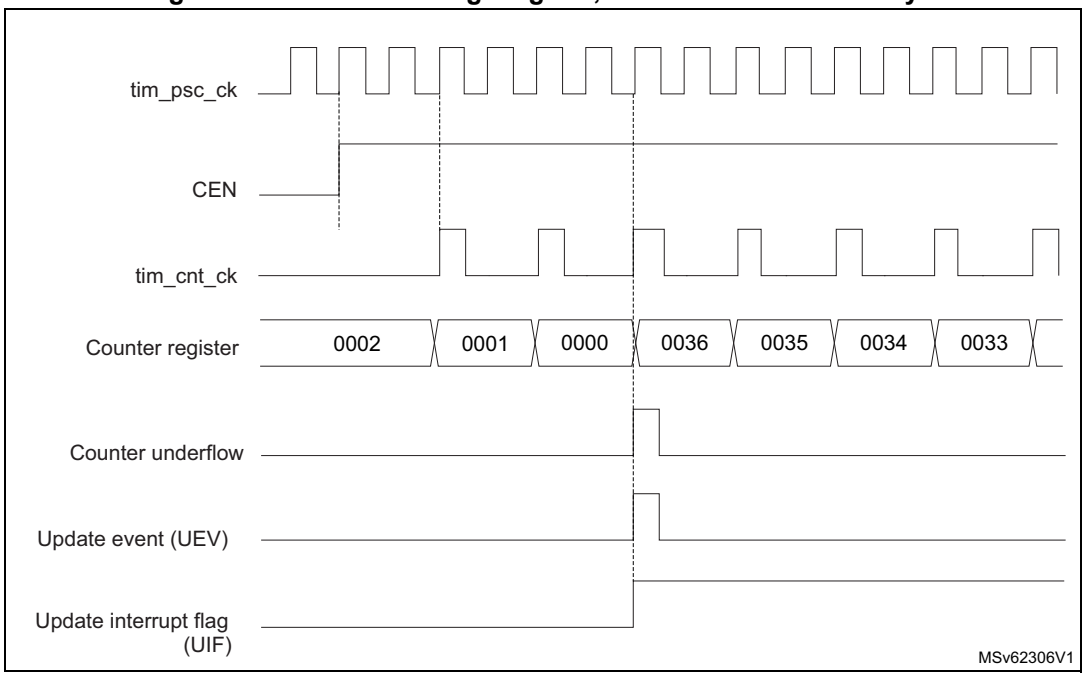


Figure 308. Counter timing diagram, internal clock divided by 4

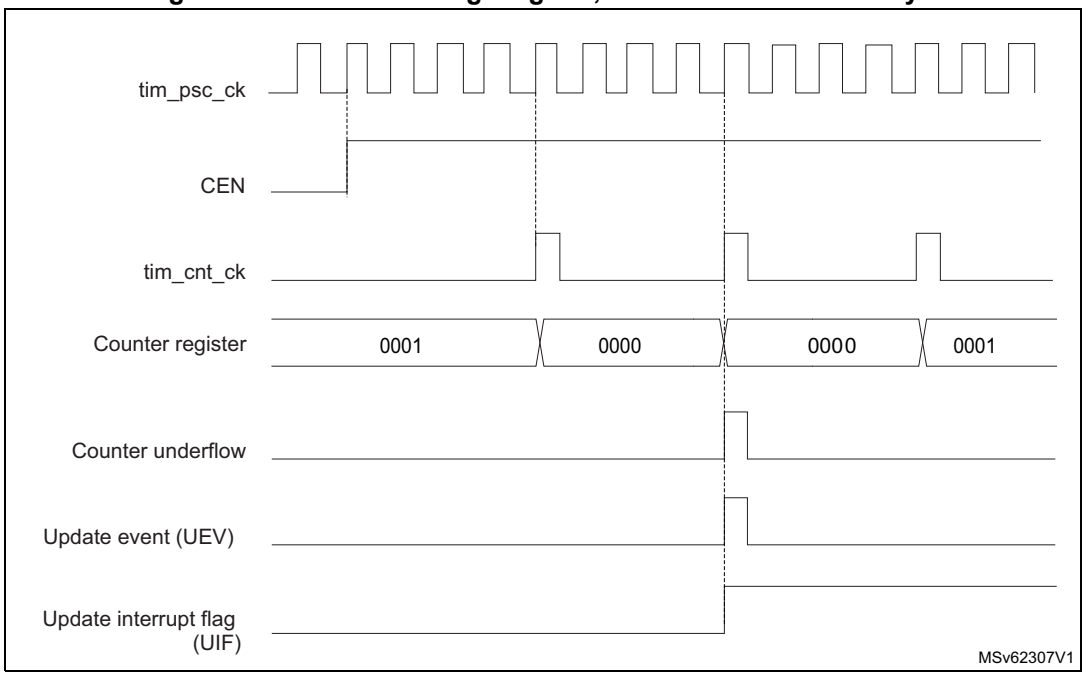


Figure 309. Counter timing diagram, internal clock divided by N

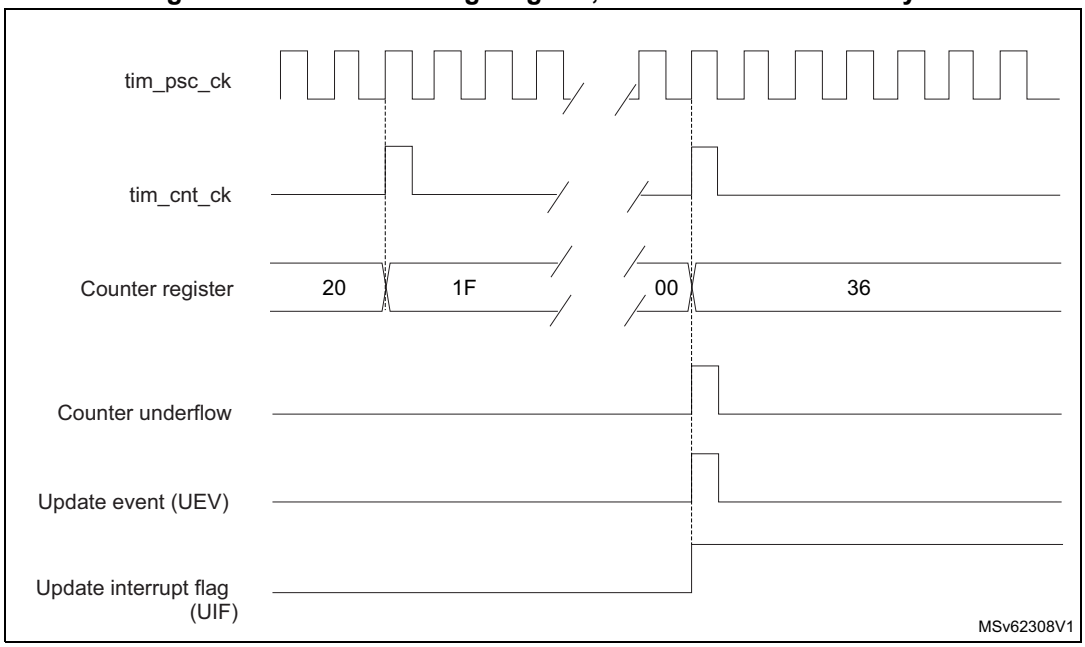
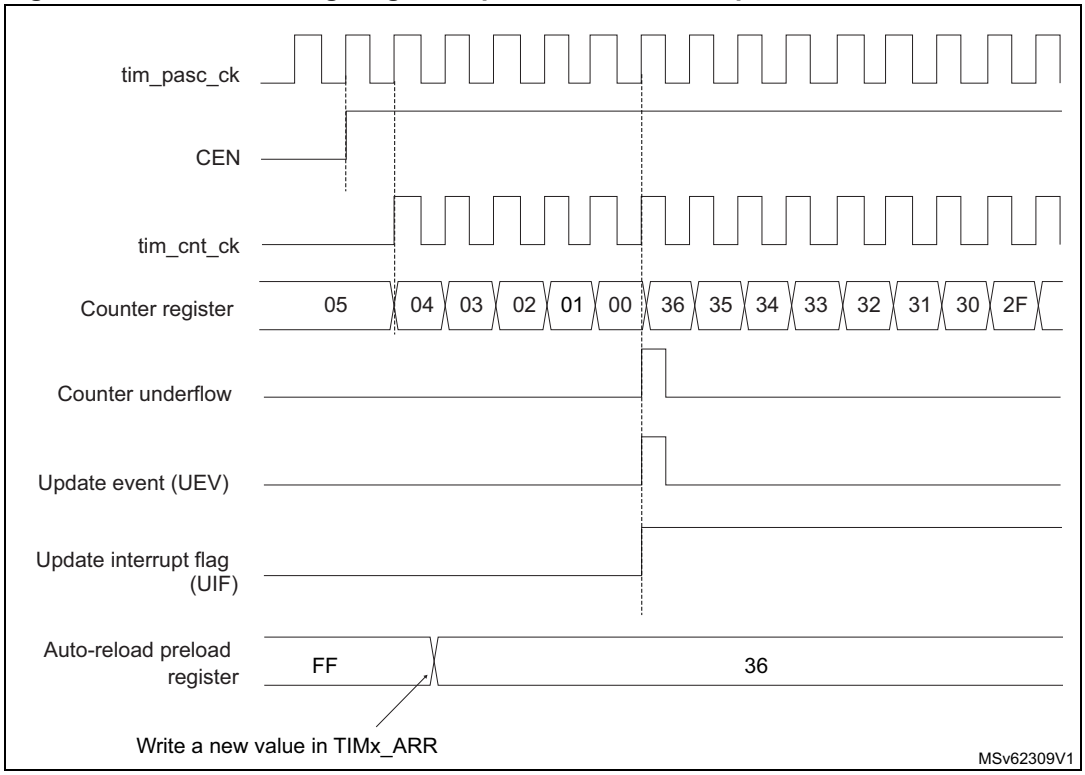


Figure 310. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

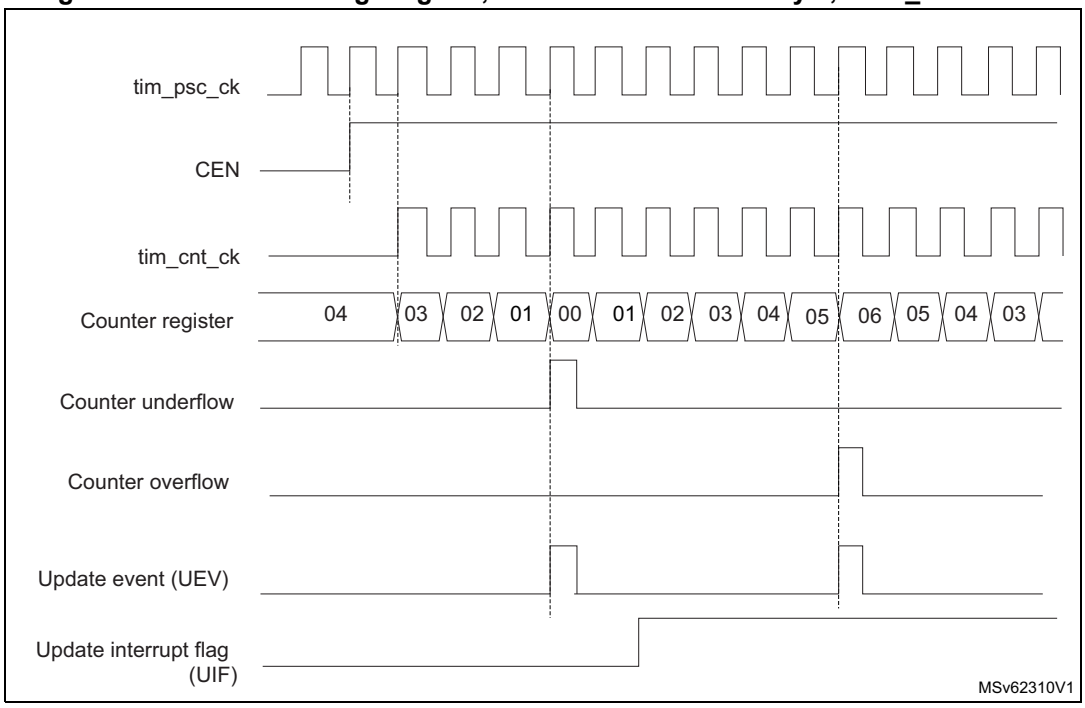
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 311. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 35.6: TIM1 registers*).

Figure 312. Counter timing diagram, internal clock divided by 2

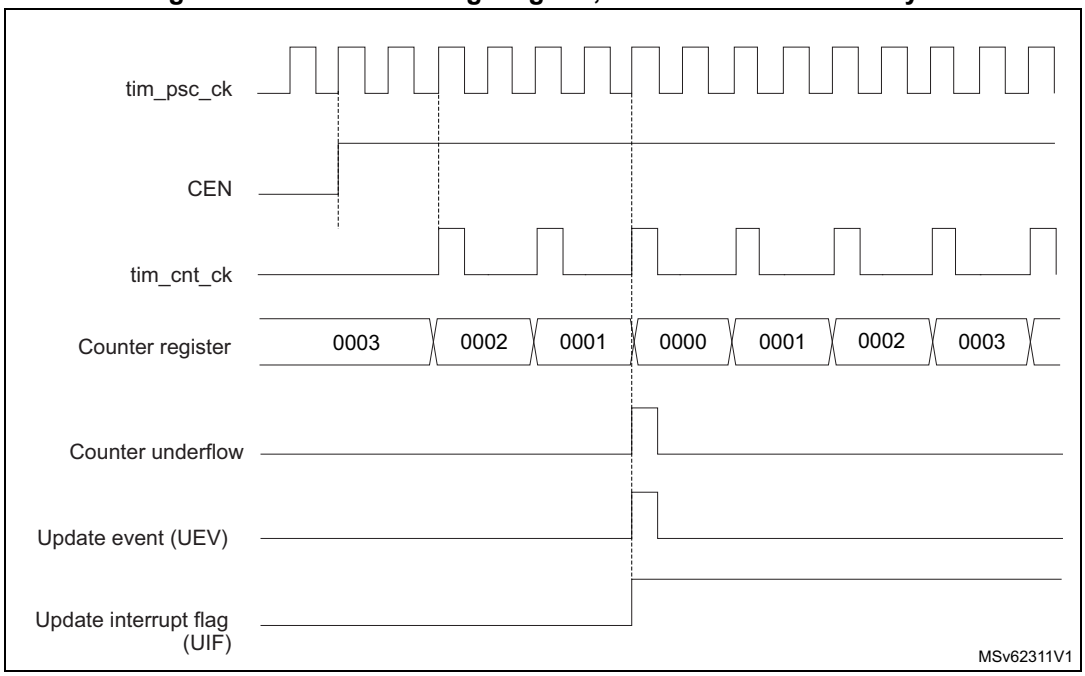


Figure 313. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

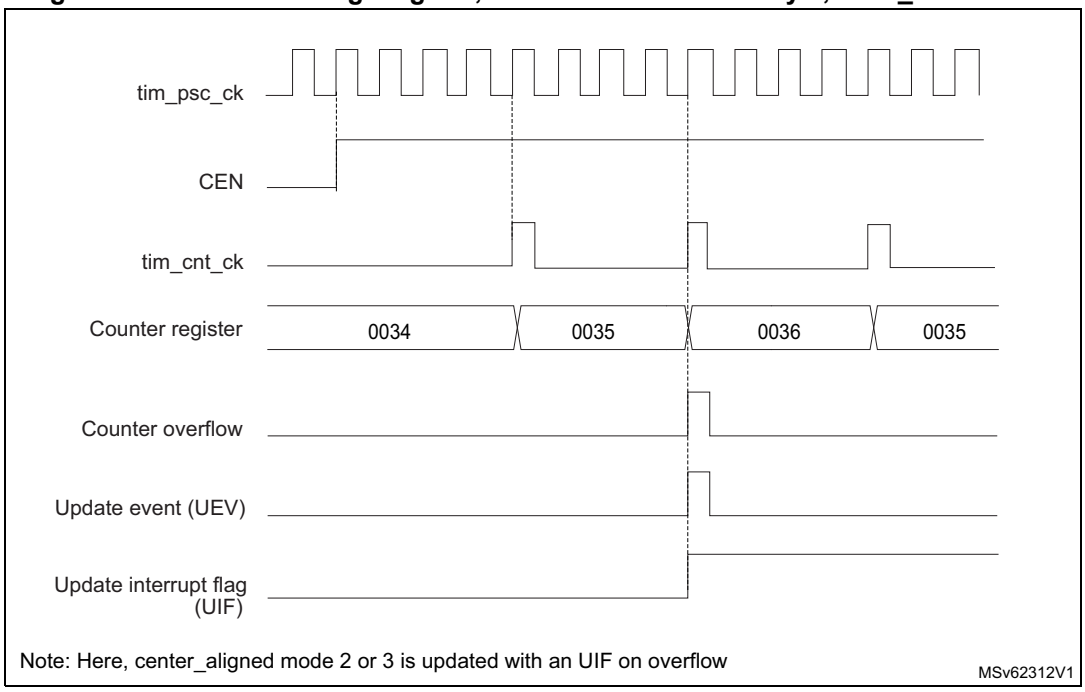


Figure 314. Counter timing diagram, internal clock divided by N

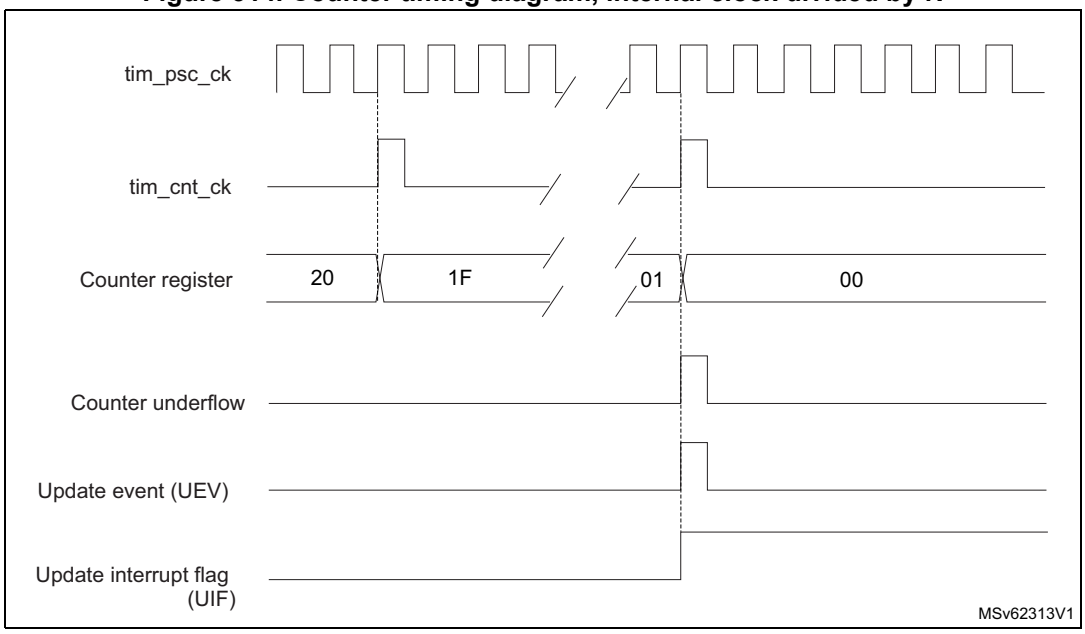


Figure 315. Counter timing diagram, update event with ARPE = 1 (counter underflow)

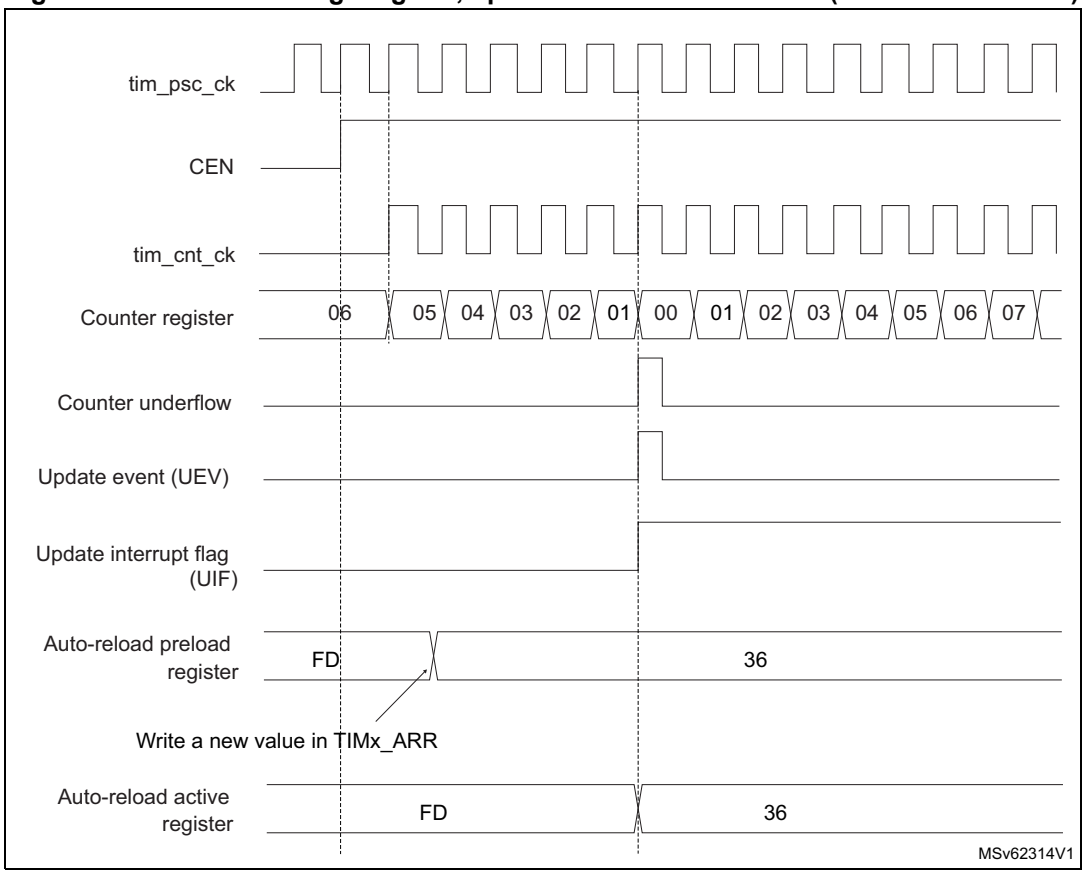
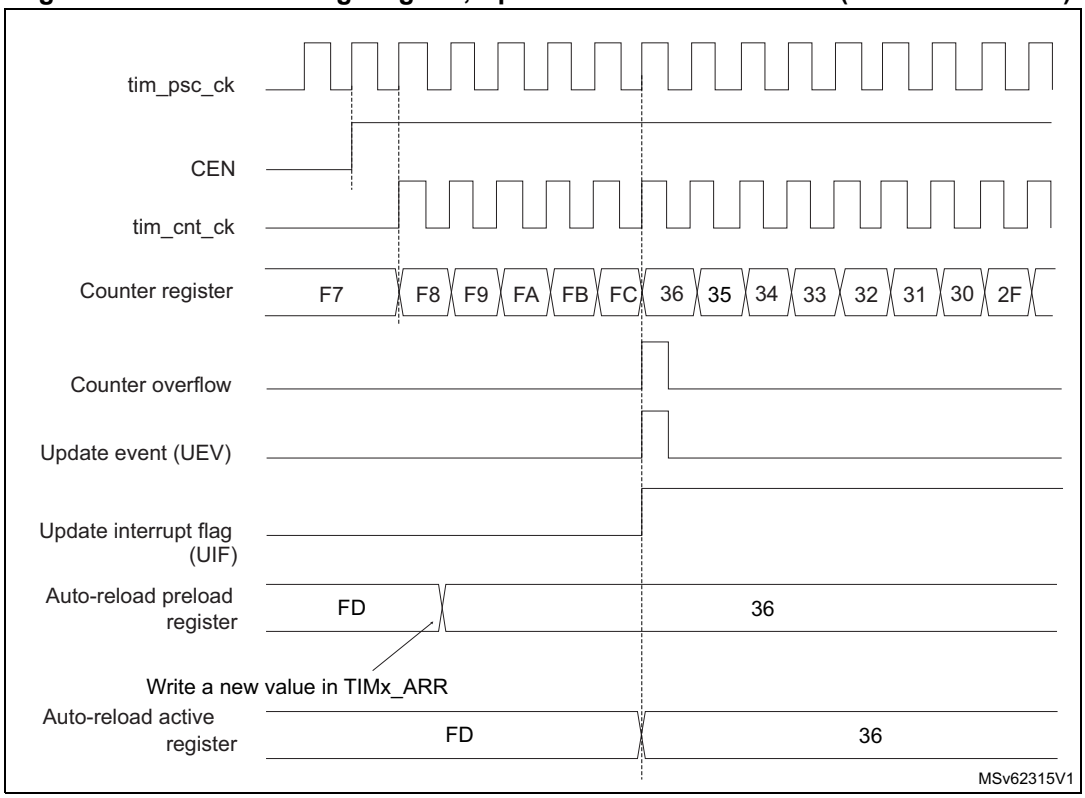


Figure 316. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



35.3.5 Repetition counter

Section 35.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

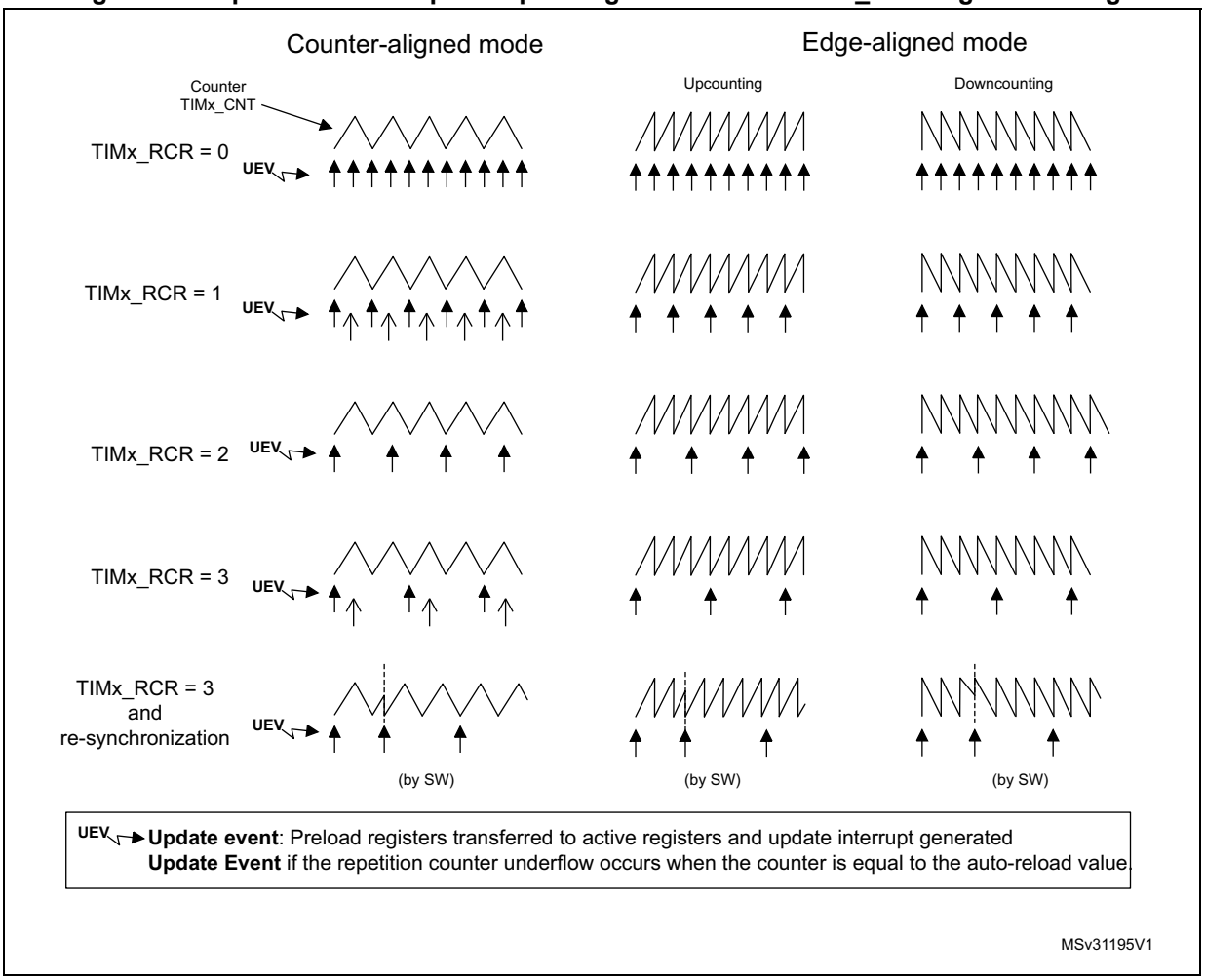
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 317*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 317. Update rate examples depending on mode and TIMx_RCR register settings



35.3.6 External trigger input

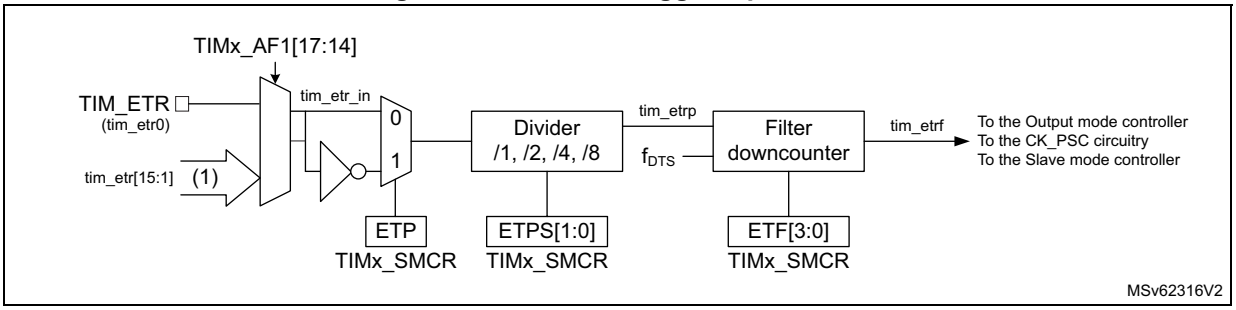
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 35.3.7*)
- trigger for the slave mode (see *Section 35.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 35.3.9*)

Figure 318 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 318. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 35.3.2: TIM1 pins and internal signals* for the list of sources connected to the etr_in input in the product.

35.3.7 Clock selection

The counter clock can be provided by the following clock sources:

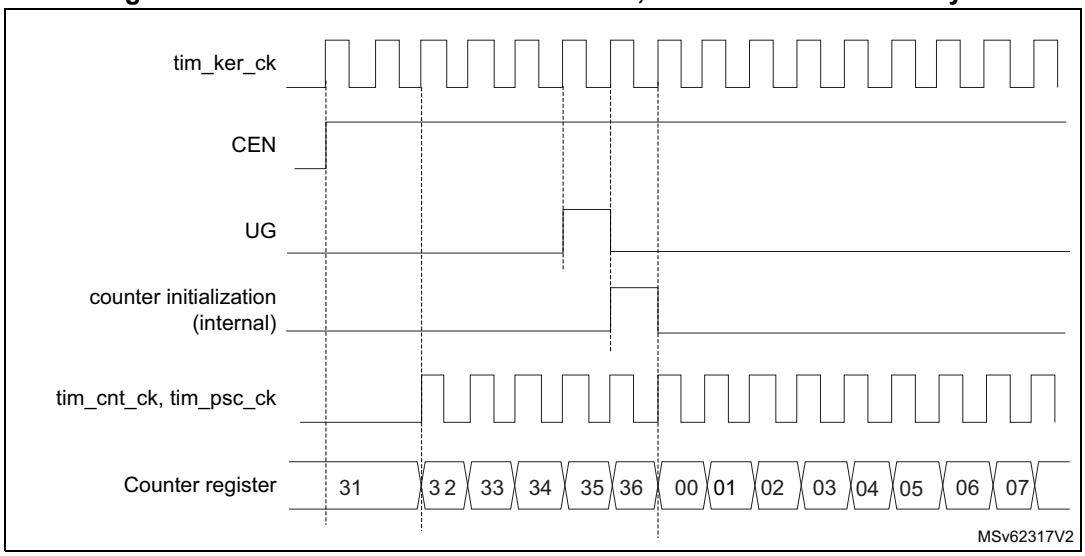
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 319 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

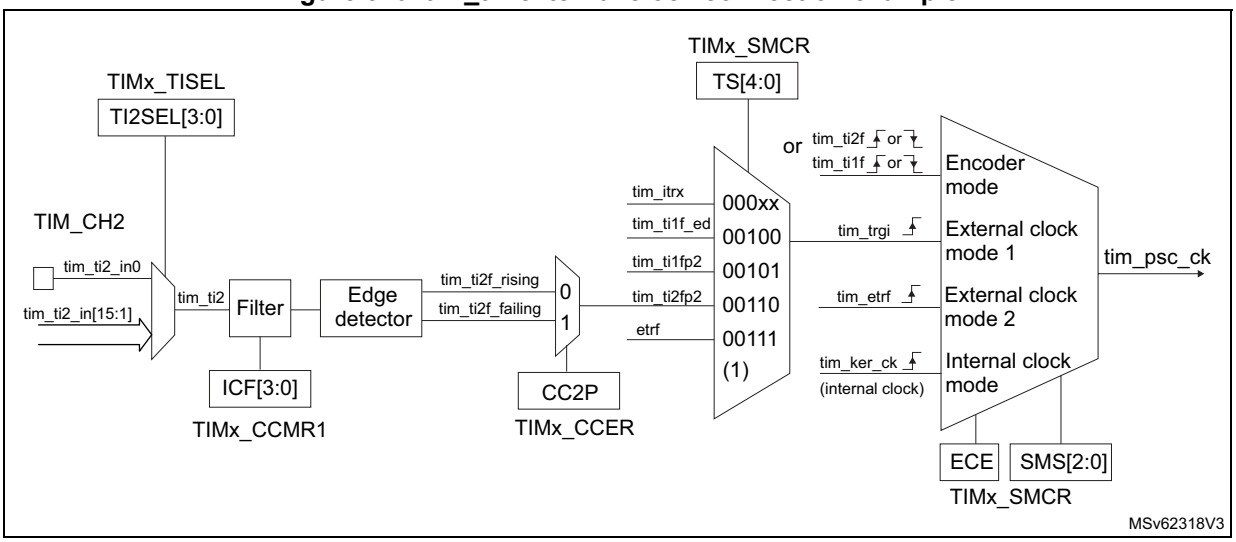
Figure 319. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 320. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

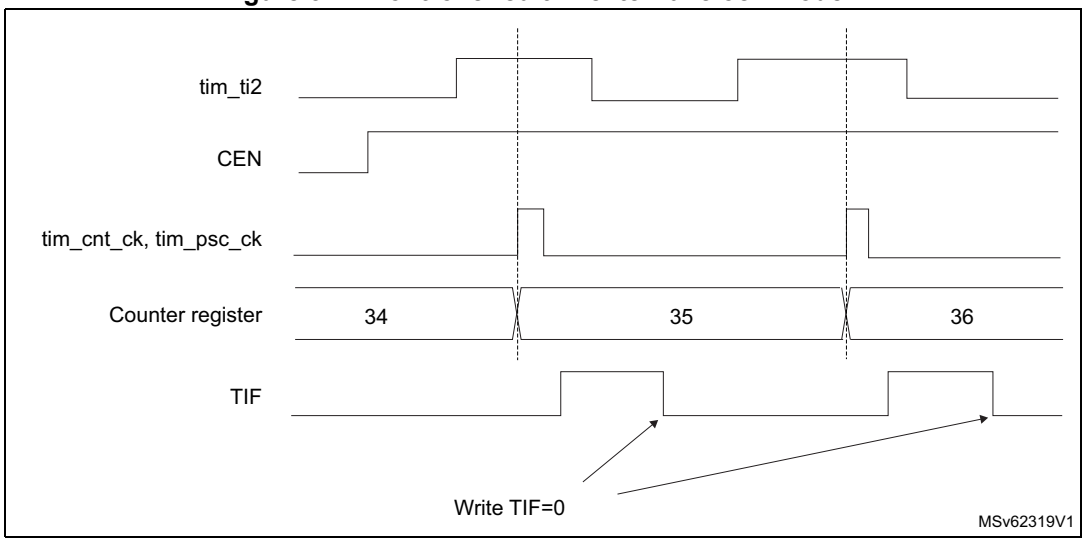
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 321. Control circuit in external clock mode 1



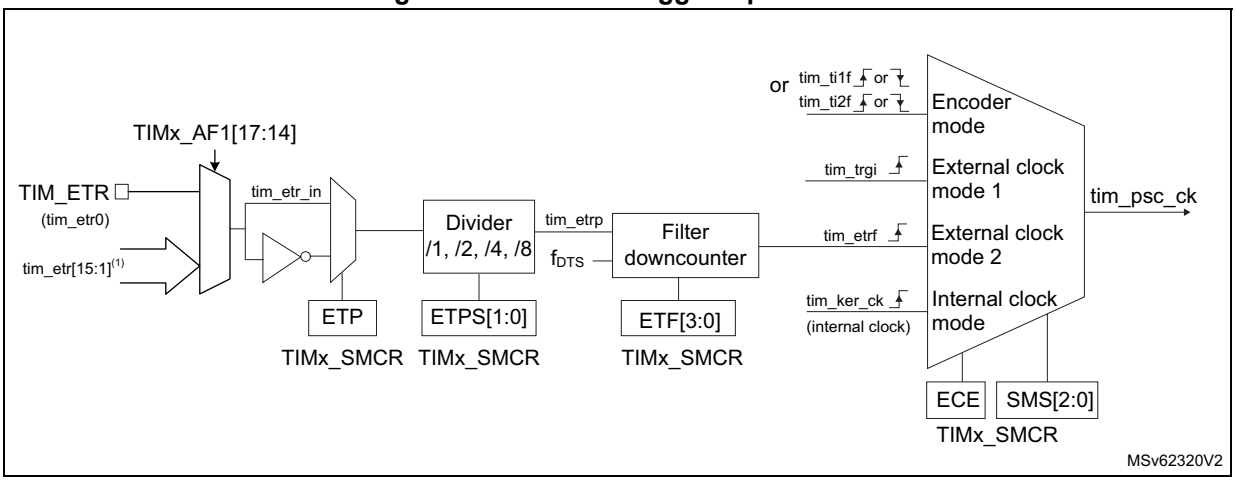
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 322* gives an overview of the external trigger input block.

Figure 322. External trigger input block



1. Refer to *Section 35.3.2: TIM1 pins and internal signals*.

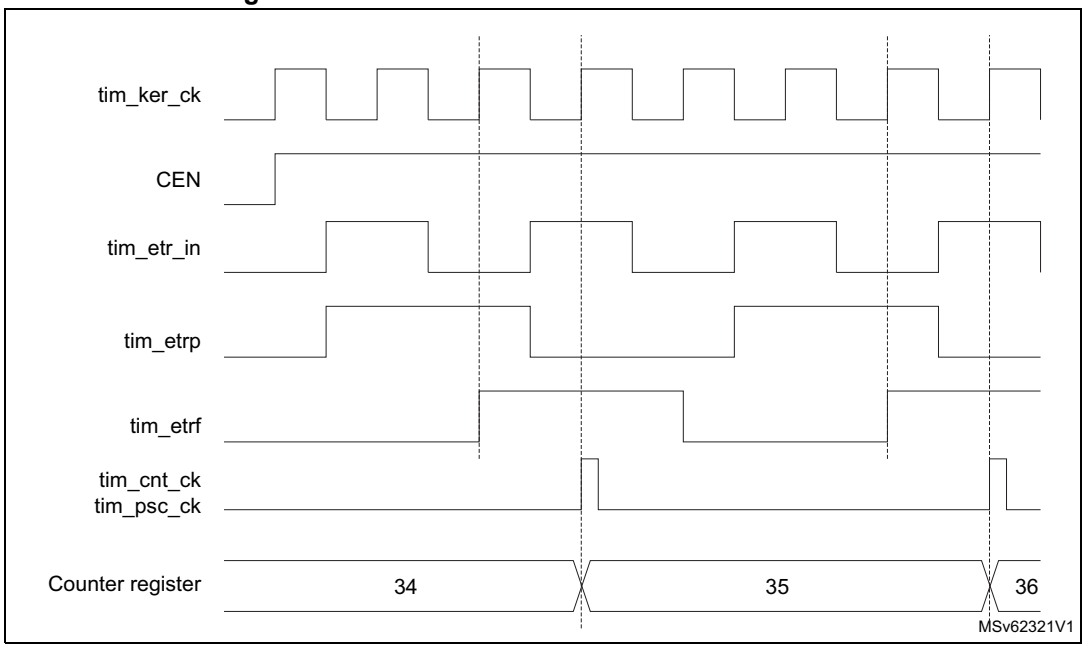
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 323. Control circuit in external clock mode 2



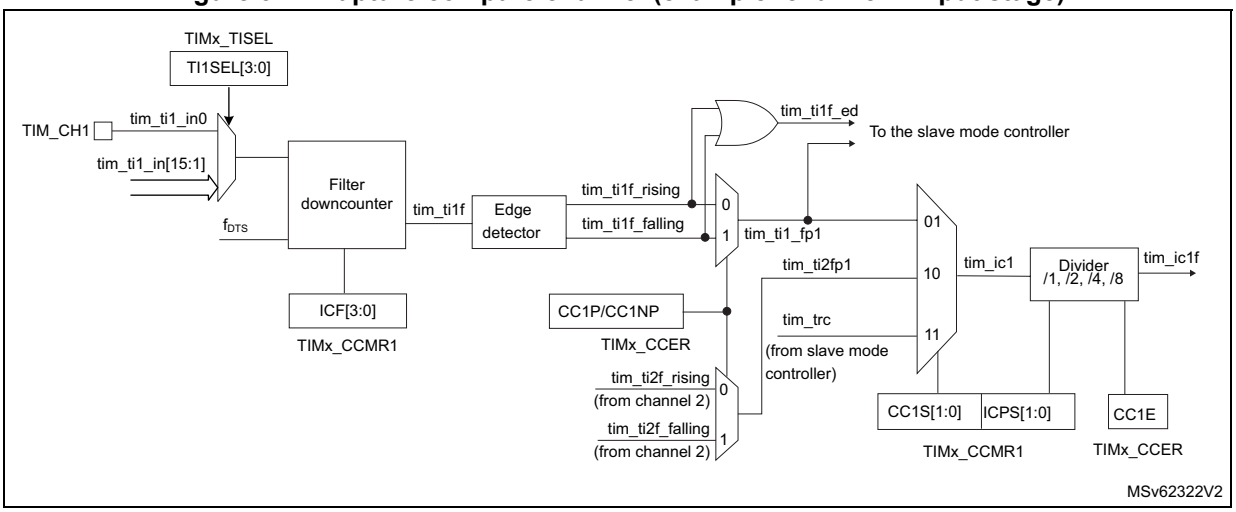
35.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 324 to *Figure 327* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 324. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 325. Capture/compare channel 1 main circuit

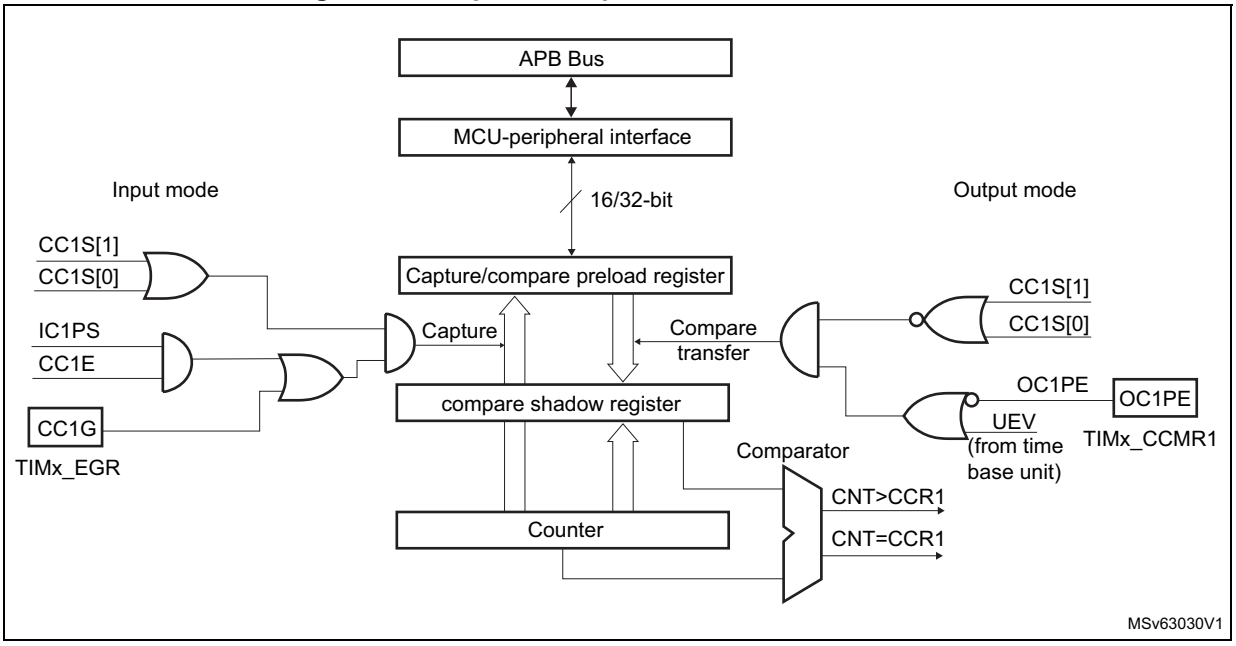
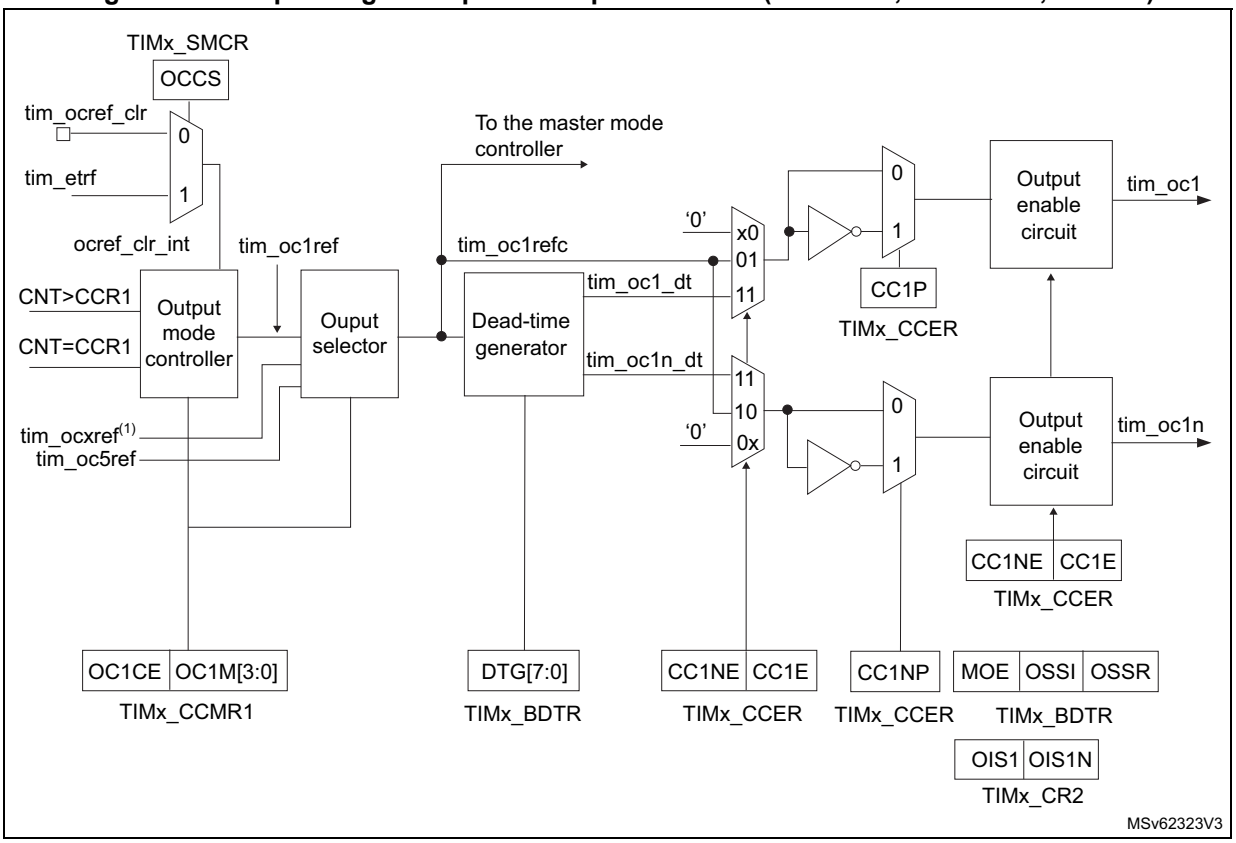
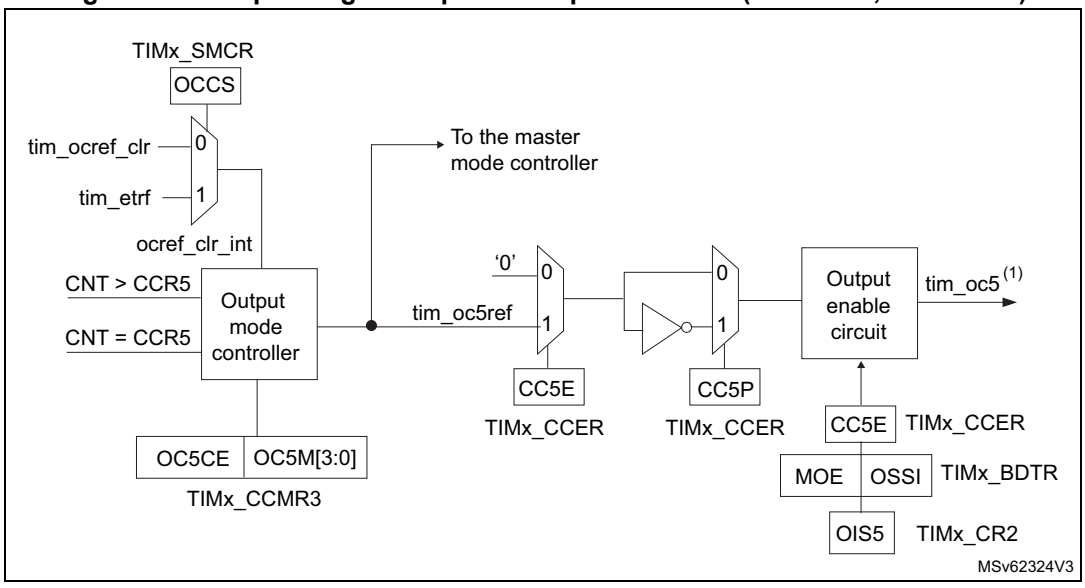


Figure 326. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 327. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

35.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

35.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

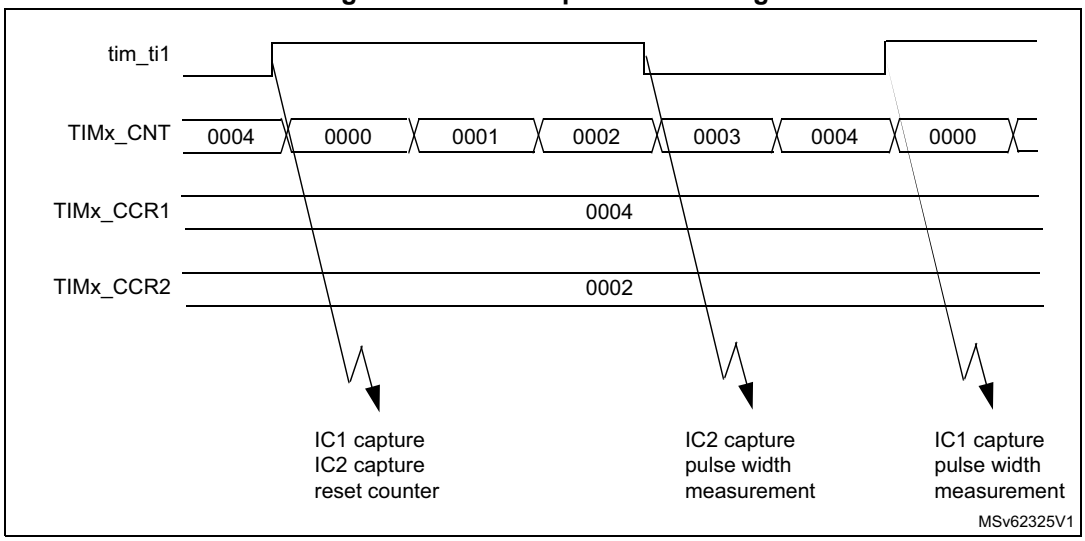
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 328. PWM input mode timing



35.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

35.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 0000), be

set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

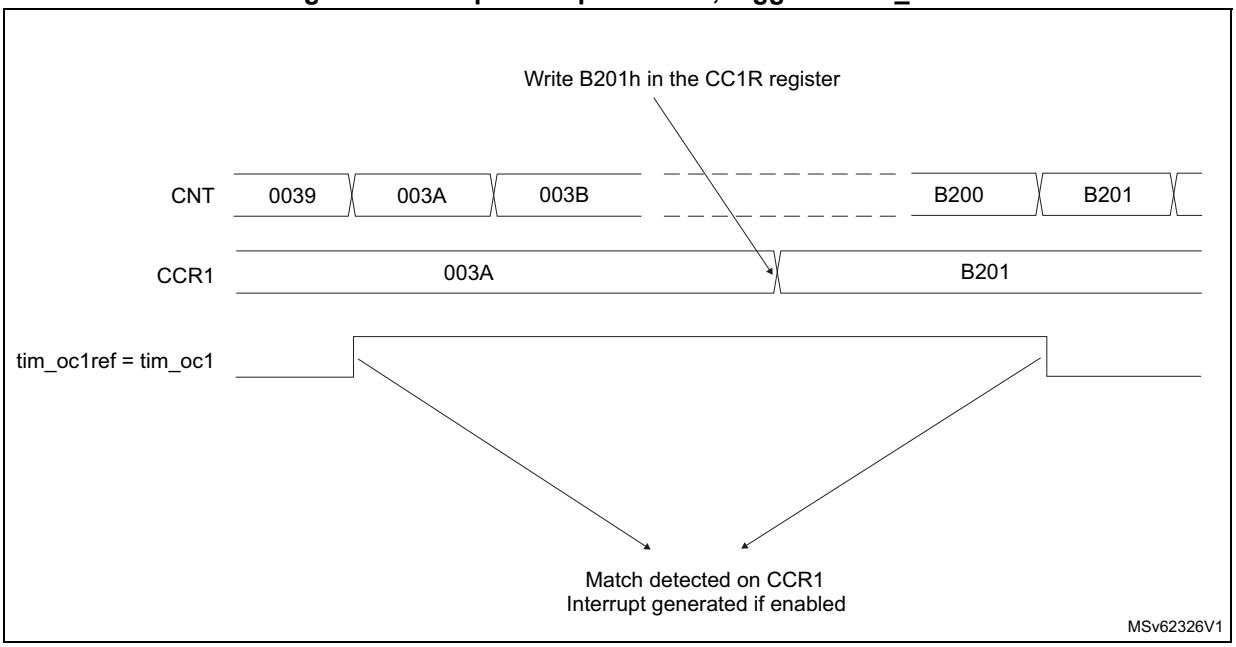
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 329*.

Figure 329. Output compare mode, toggle on tim_oc1



35.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

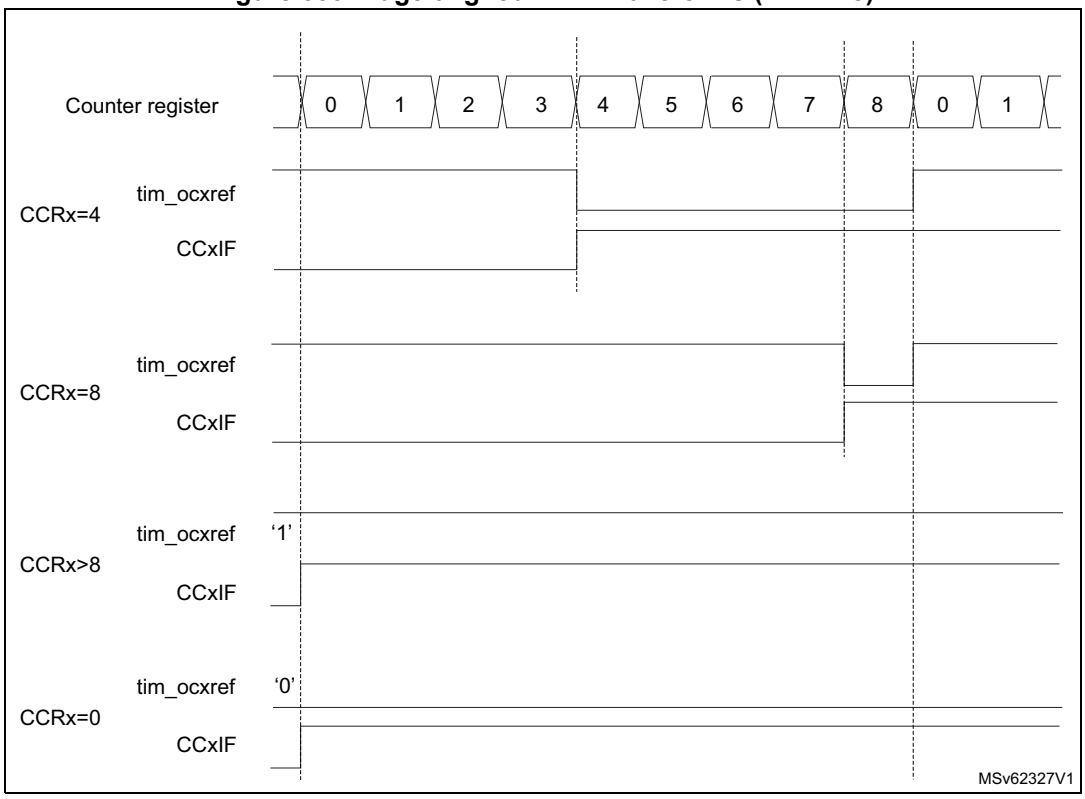
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 330* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 330. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

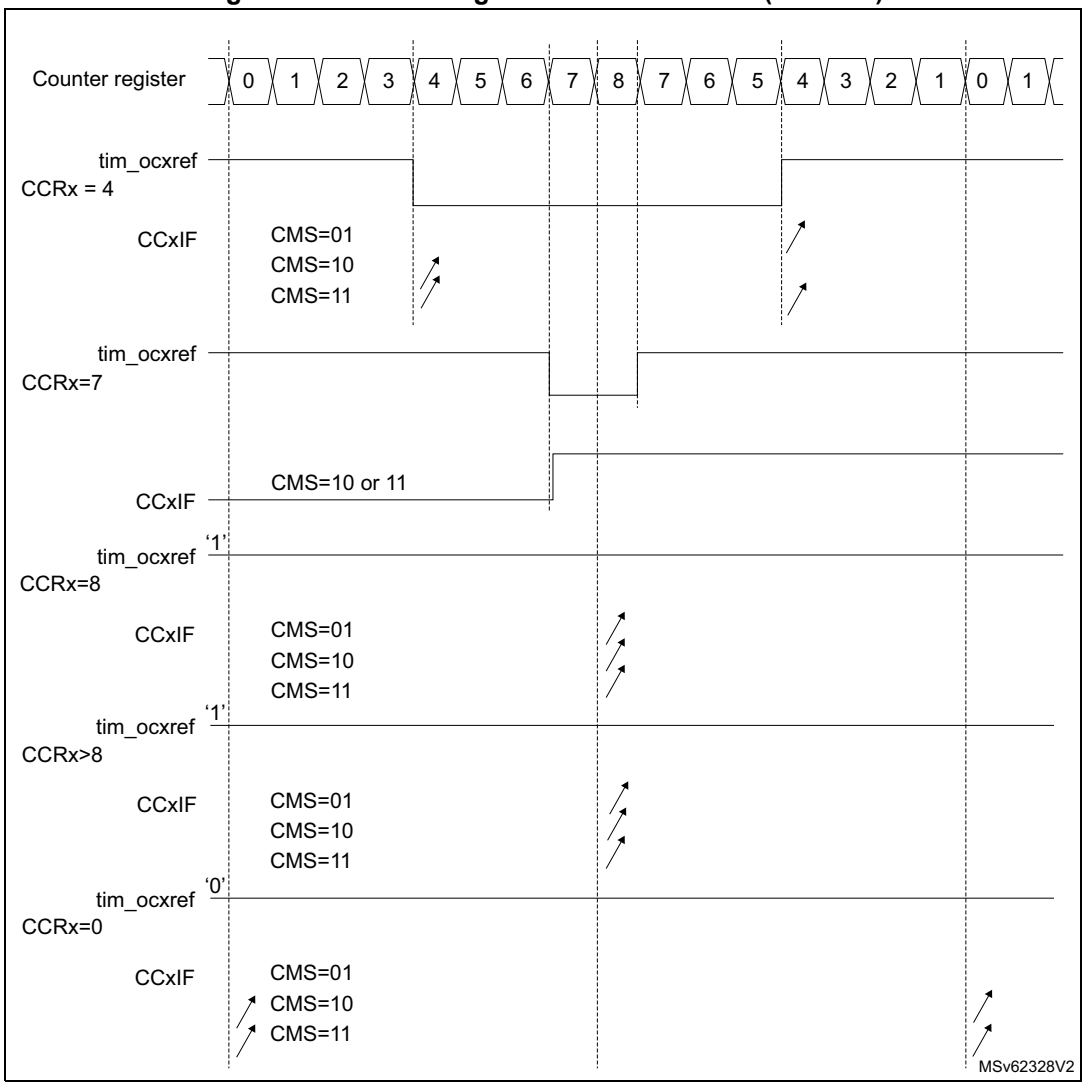
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 331 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 331. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

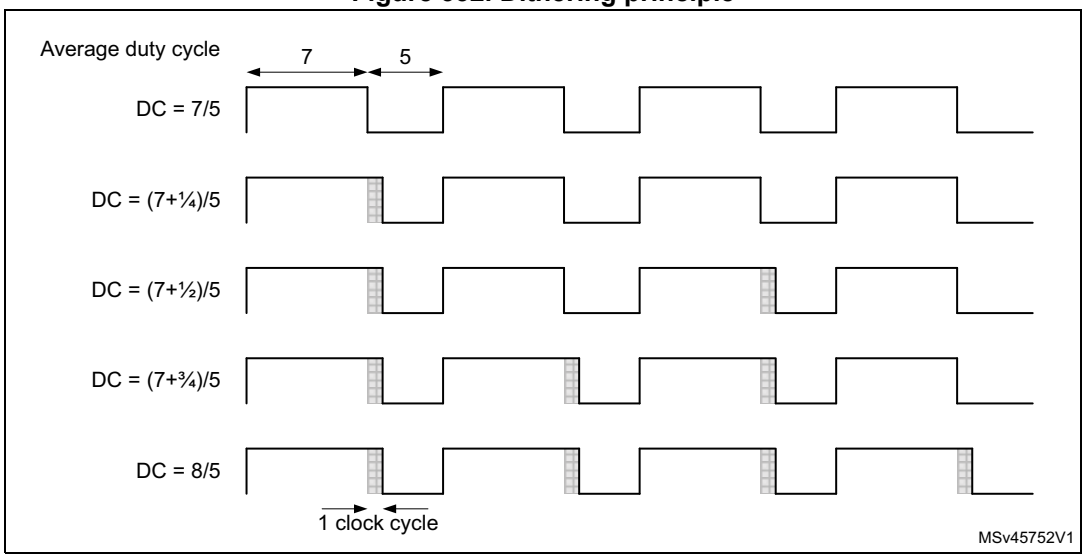
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 332* presents the dithering principle applied to four consecutive PWM cycles.

Figure 332. Dithering principle

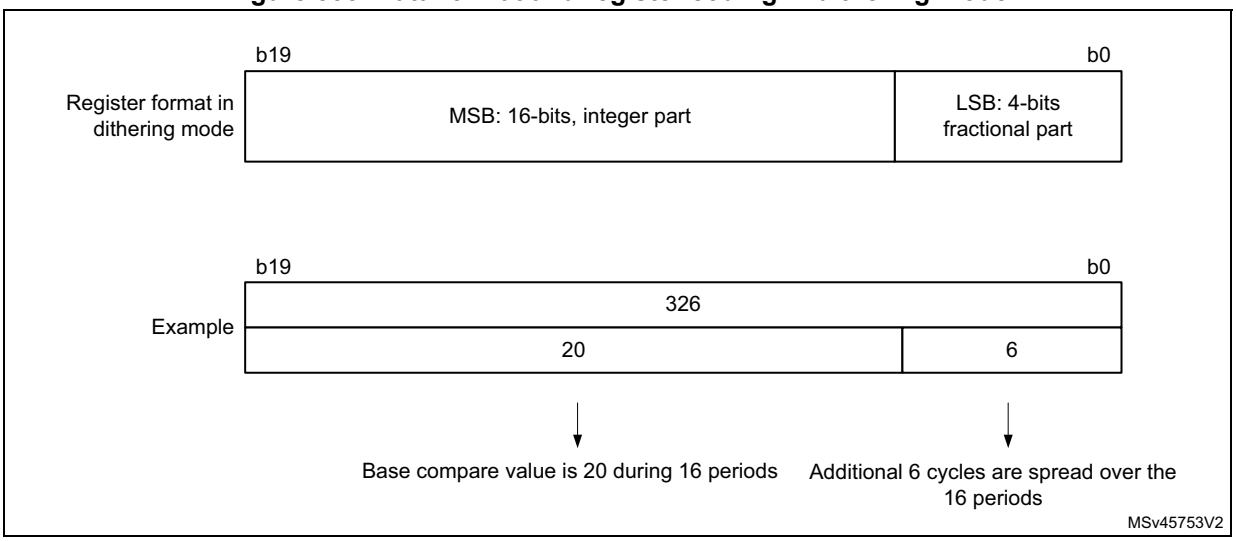


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 333* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 333. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

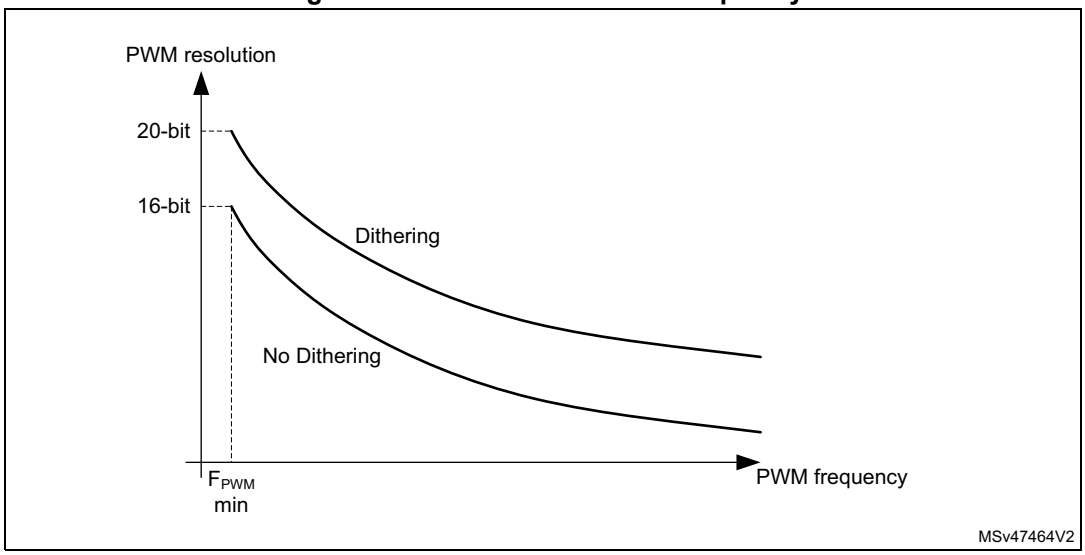
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

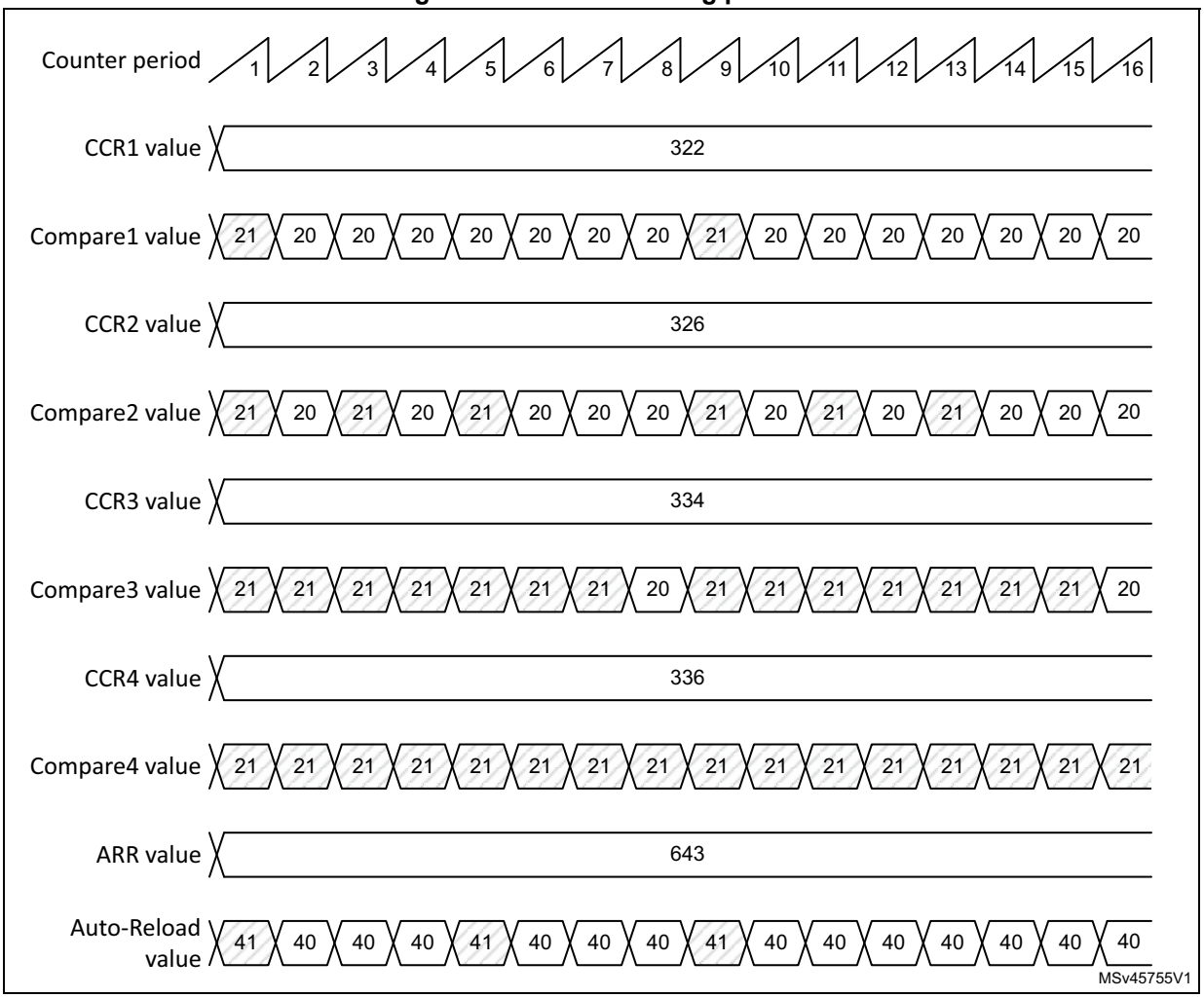
As shown on *Figure 334*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 334. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 335*.

Figure 335. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 353*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 353. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 353. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 336*.

Figure 336. Dithering effect on duty cycle in center-aligned PWM mode

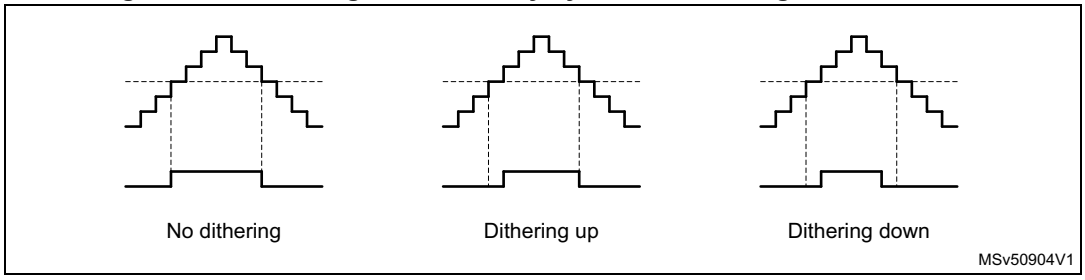


Table 354 shows how the dithering pattern is added in center-aligned PWM mode.

Table 354. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 354. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

35.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

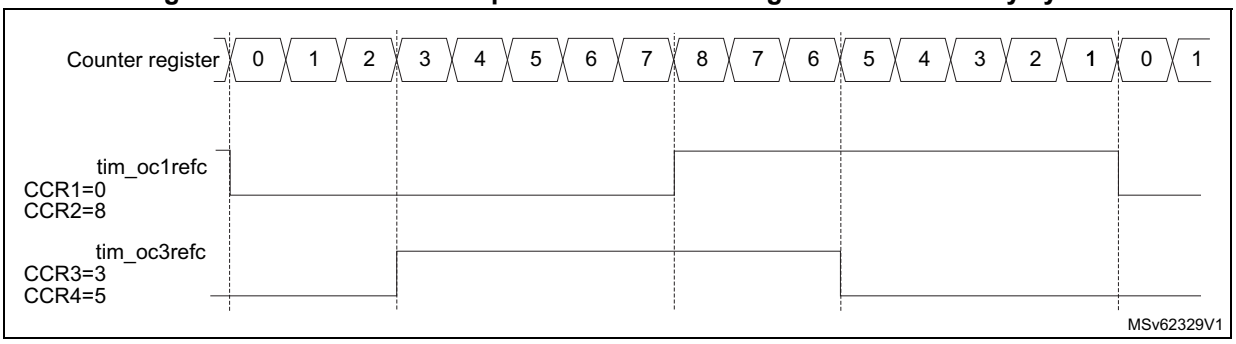
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 337 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 337. Generation of 2 phase-shifted PWM signals with 50% duty cycle



35.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

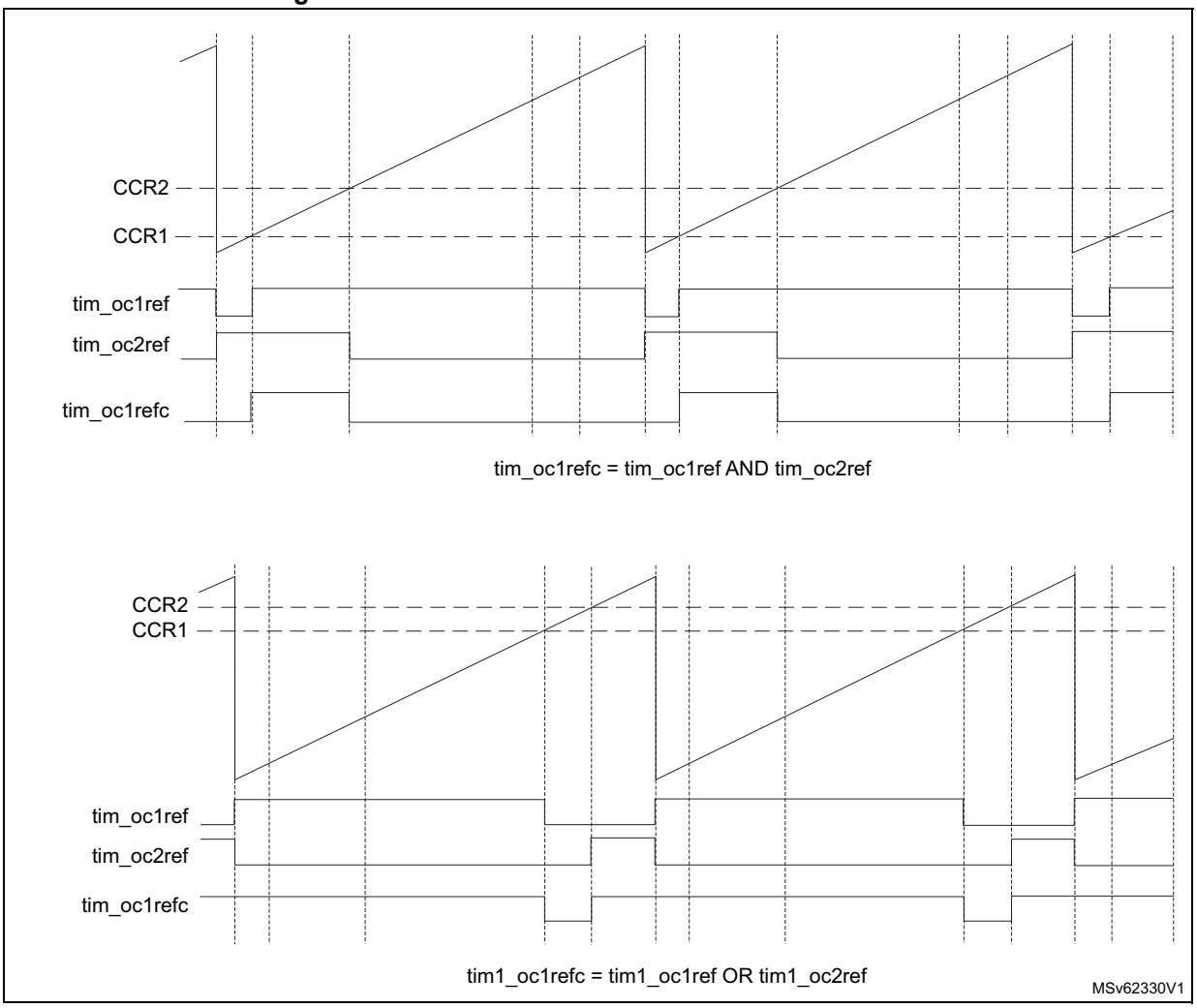
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 338 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 338. Combined PWM mode on channel 1 and 3



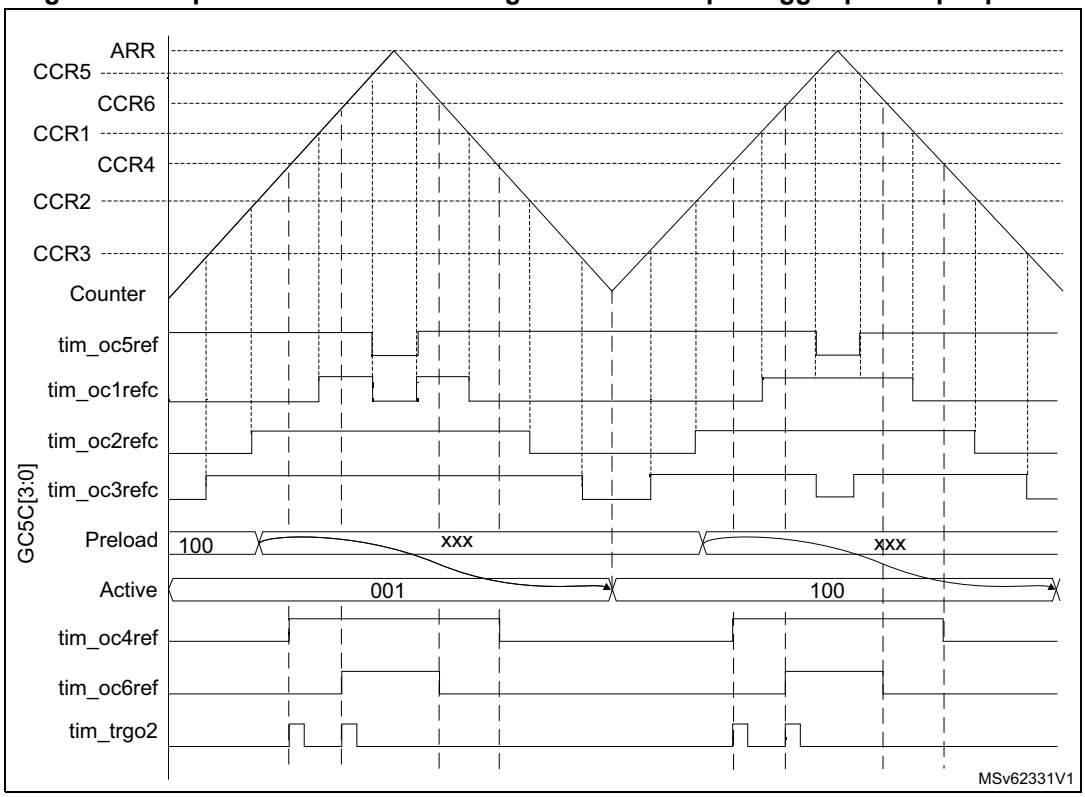
35.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 339. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 35.3.31: ADC triggers* for more details.

35.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 362: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

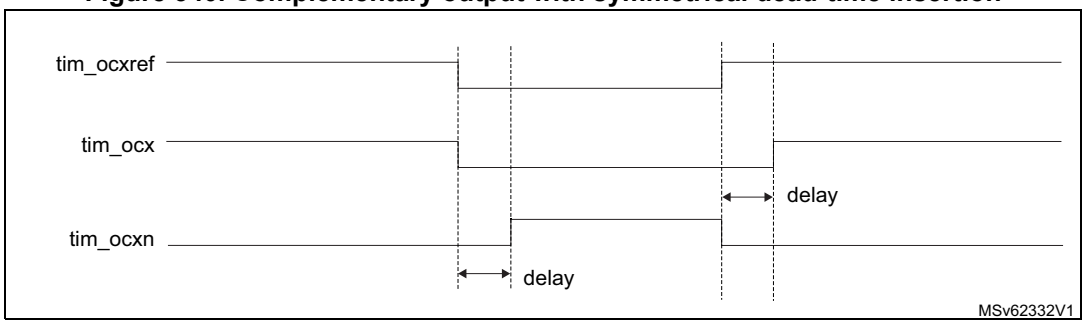
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 340. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 341*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 341. Asymmetrical deadtime

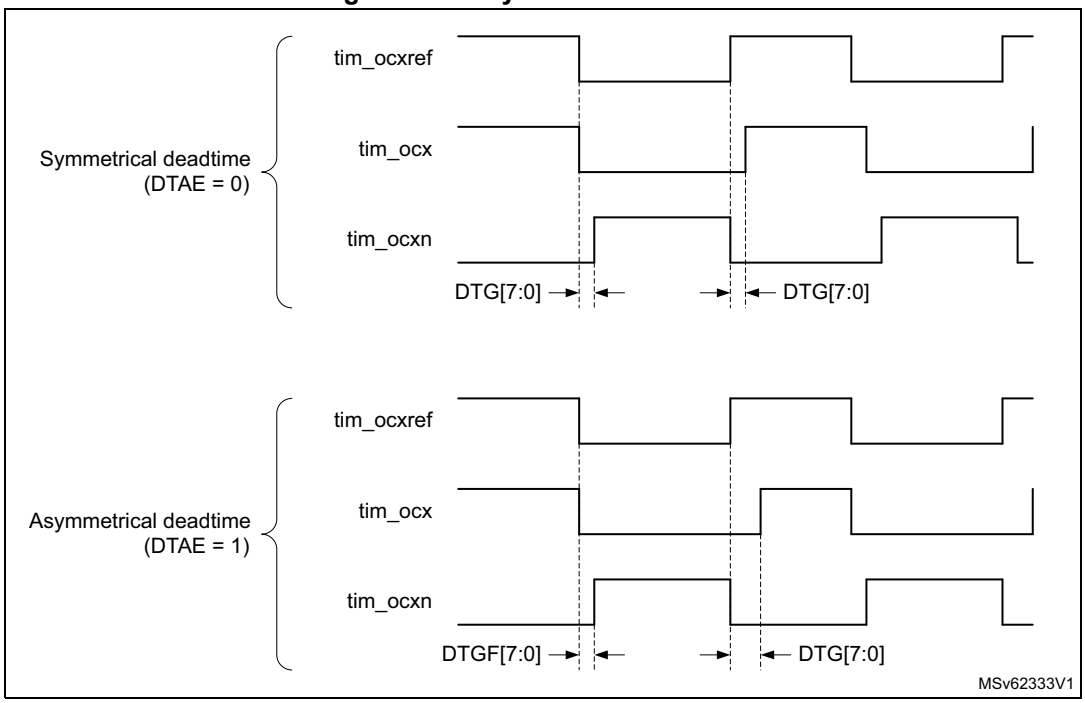


Figure 342. Dead-time waveforms with delay greater than the negative pulse

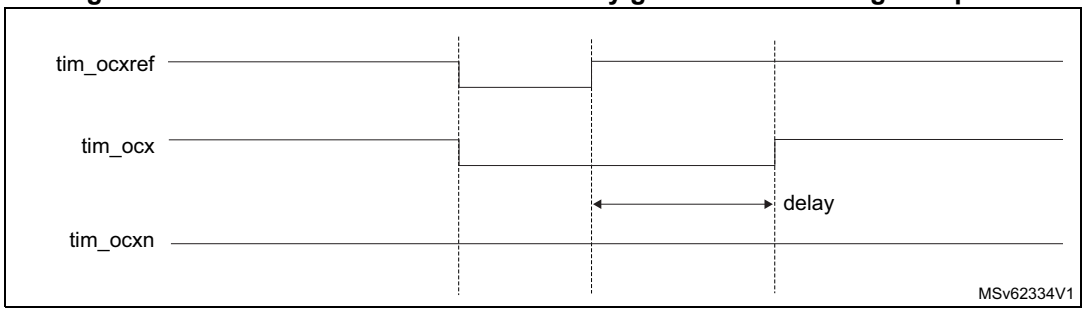
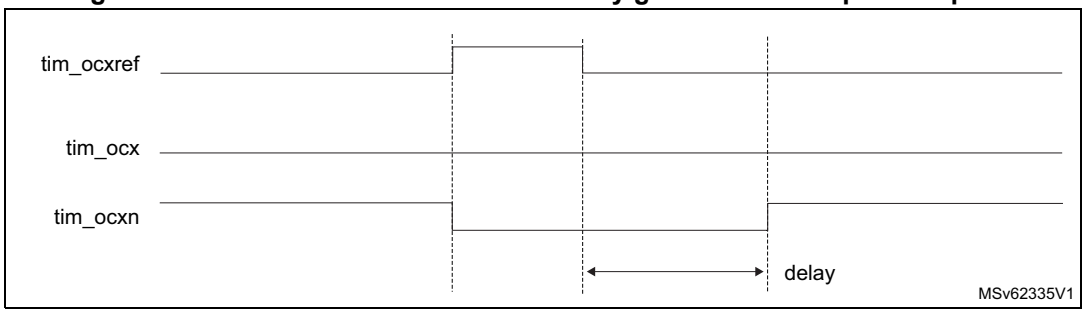


Figure 343. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 35.6.20: TIM1 break and dead-time register (TIM1_BDTR)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

35.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 362: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 35.3.2: TIM1 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 35.3.2: TIM1 pins and internal signals* for product specific implementation)

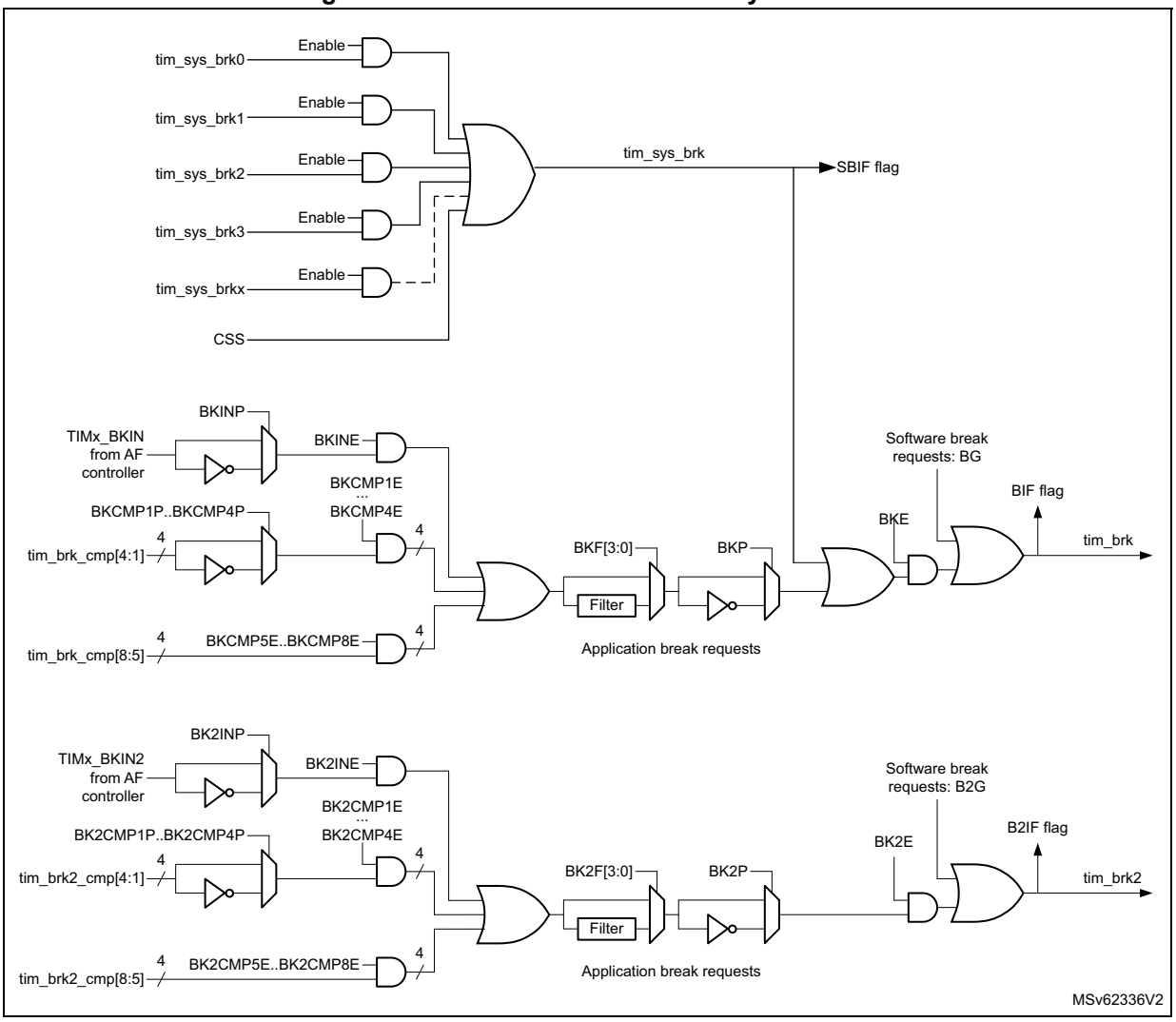
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 35.3.2: TIM1 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 344* below.

Figure 344. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

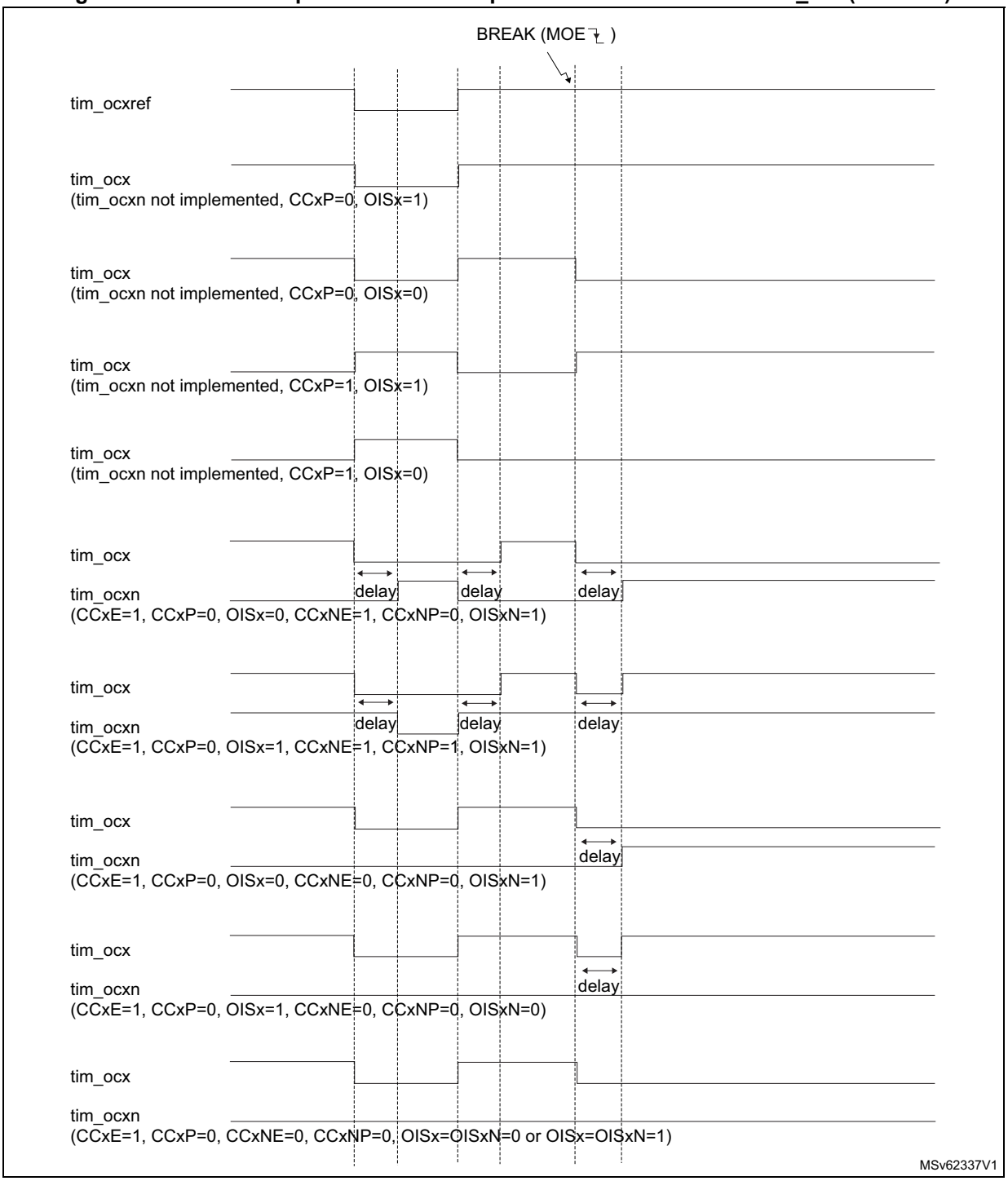
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 35.6.20*. The LOCK bits can be written only once after an MCU reset.

Figure 345 shows an example of behavior of the outputs in response to a break.

Figure 345. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 355*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 355. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 346 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 346. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

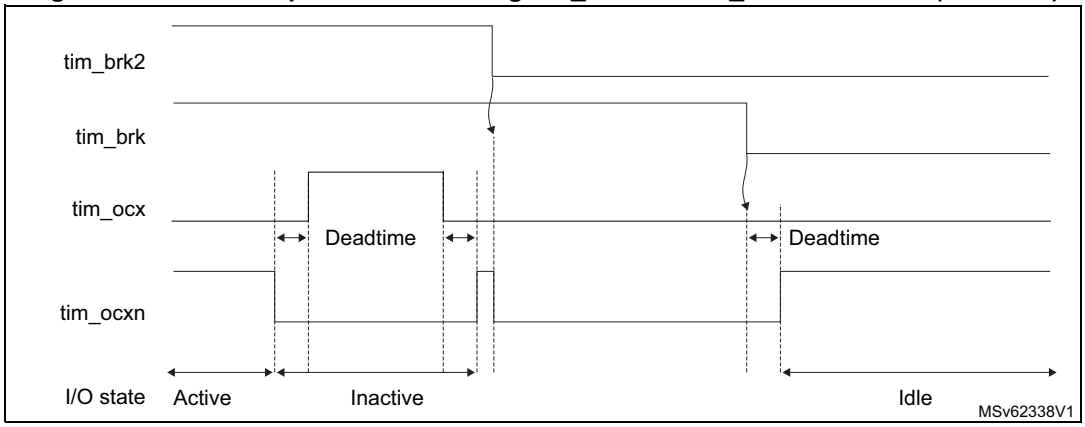
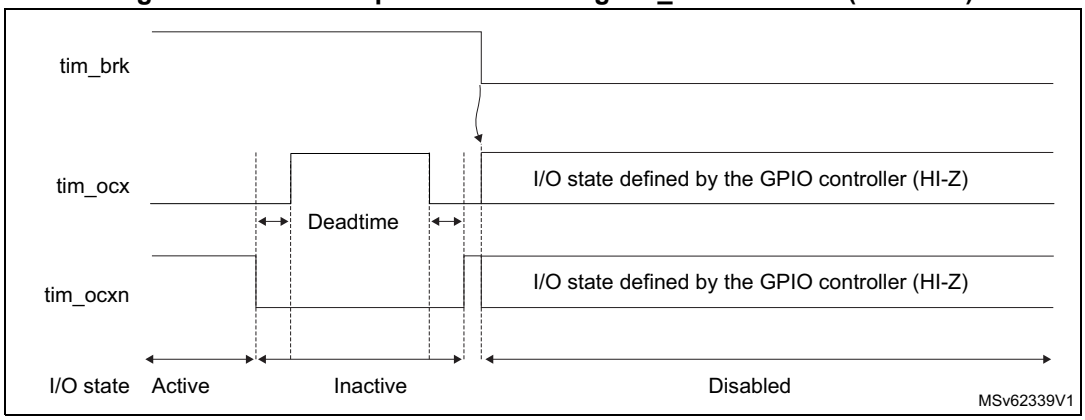


Figure 347. PWM output state following tim_brk assertion (OSSI = 0)



35.3.19 Bidirectional break inputs

The TIM1 features bidirectional break I/Os, as represented on *Figure 348*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 356*).

Table 356. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

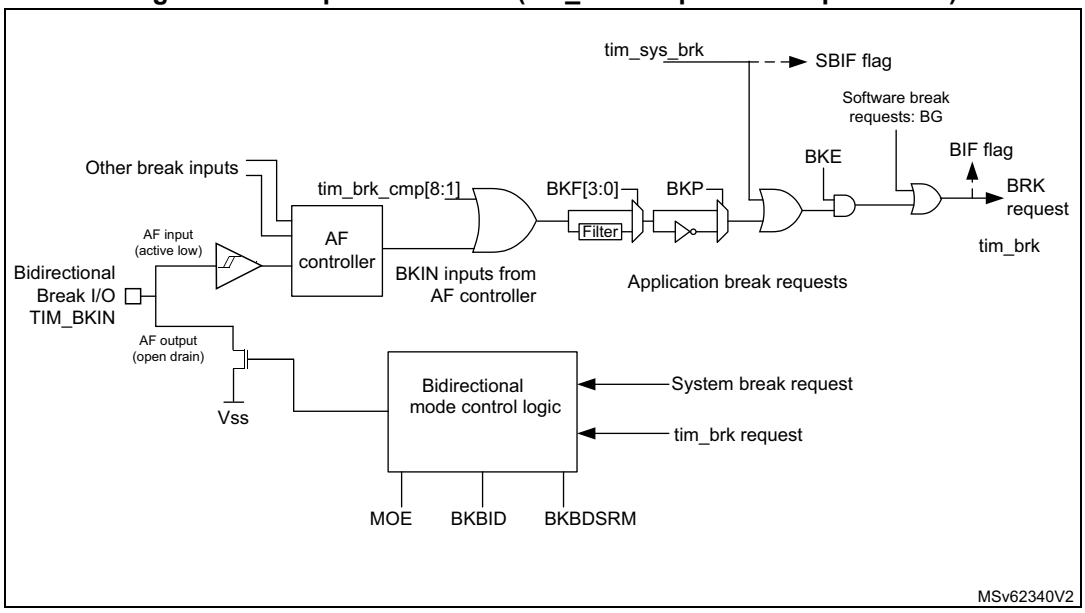
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 348. Output redirection (tim_brk2 request not represented)



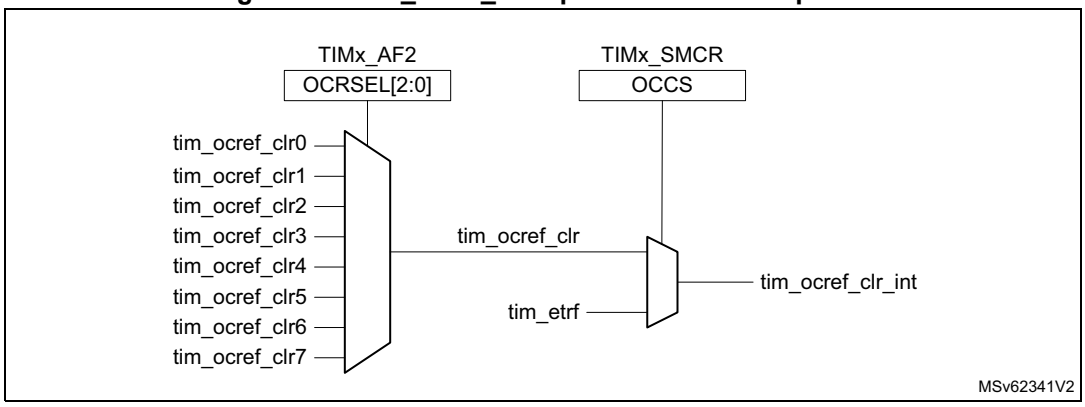
35.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM

cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the *Figure 349* below. Refer to *Section 35.3.2: TIM1 pins and internal signals* for a list of sources available in the product.

Figure 349. tim_ocref_clr input selection multiplexer

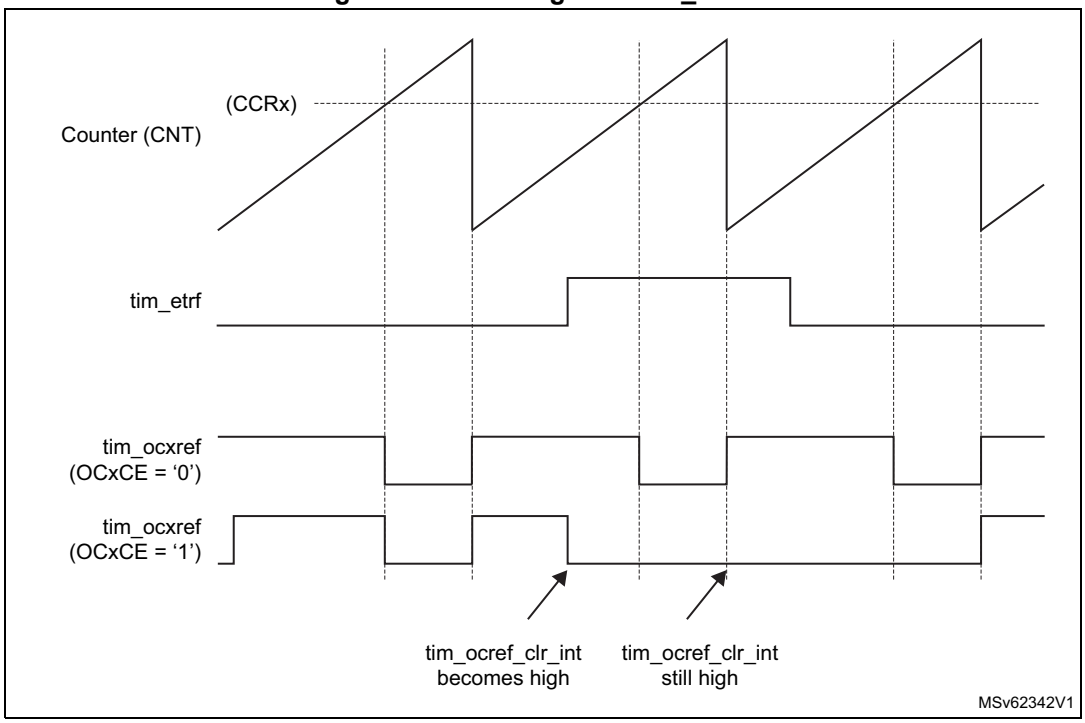


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 350 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 350. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

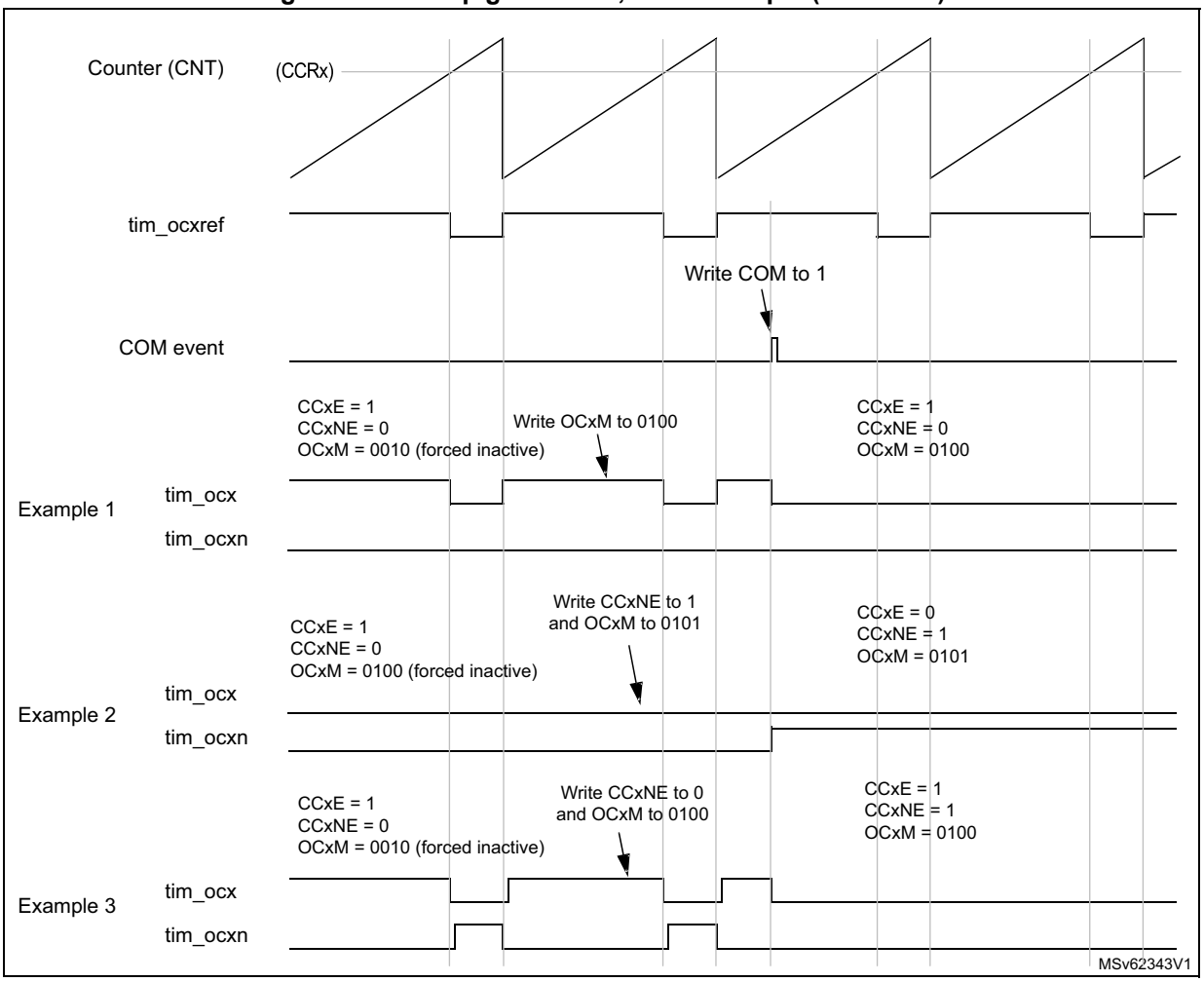
35.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 351 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 351. 6-step generation, COM example (OSSR = 1)



35.3.22 One-pulse mode

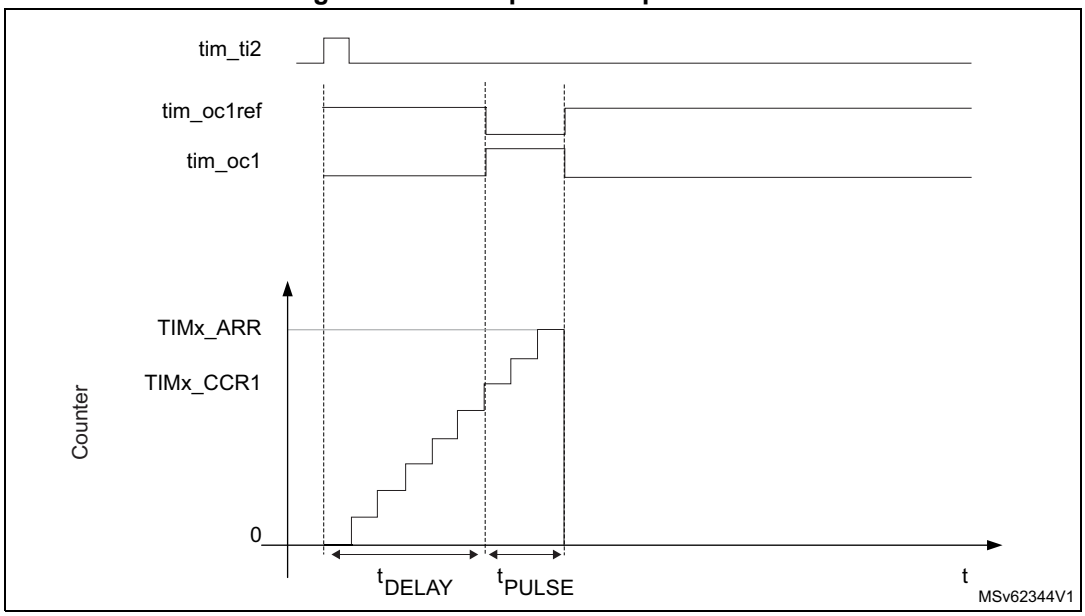
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 352. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

35.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 35.3.22*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

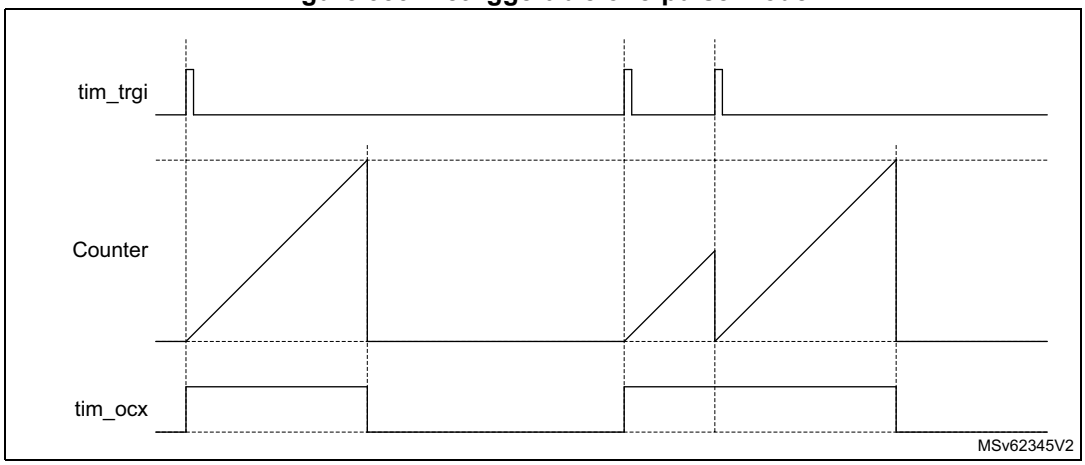
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 353. Retriggerable one-pulse mode



35.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 354*.

Figure 354. Pulse generator circuitry

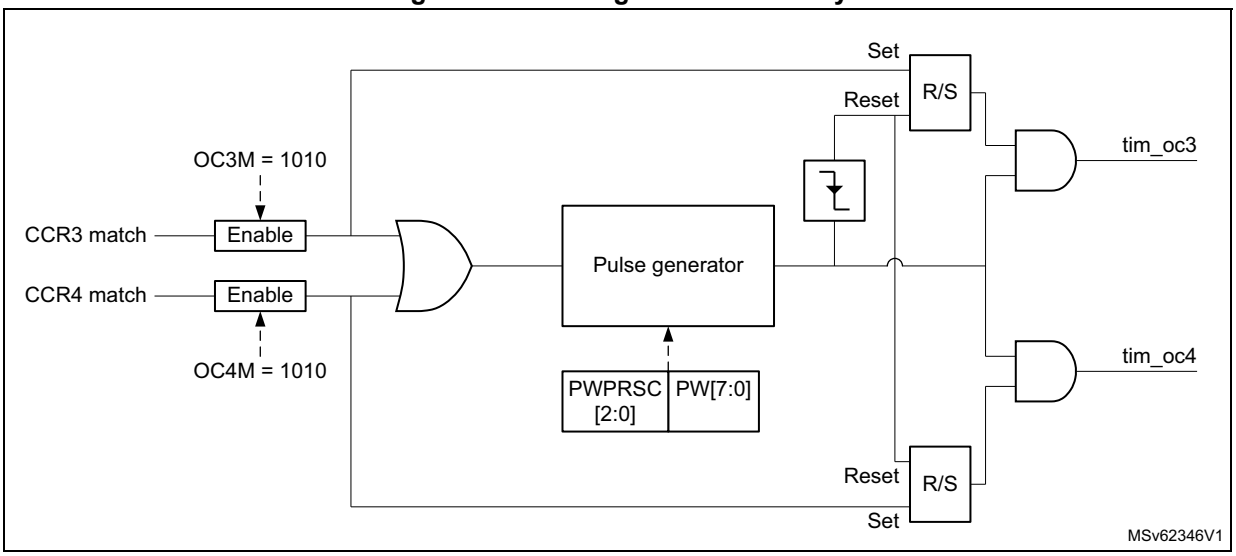
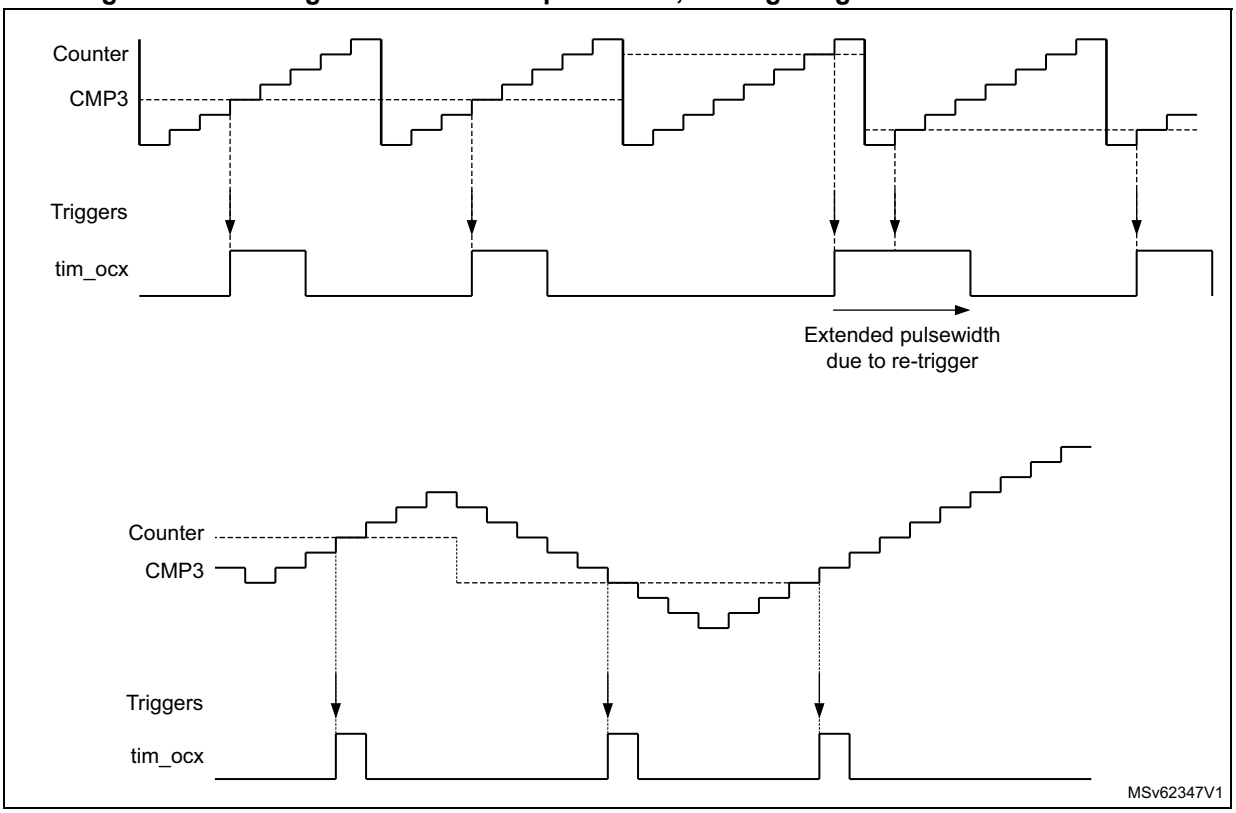


Figure 355 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 355. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

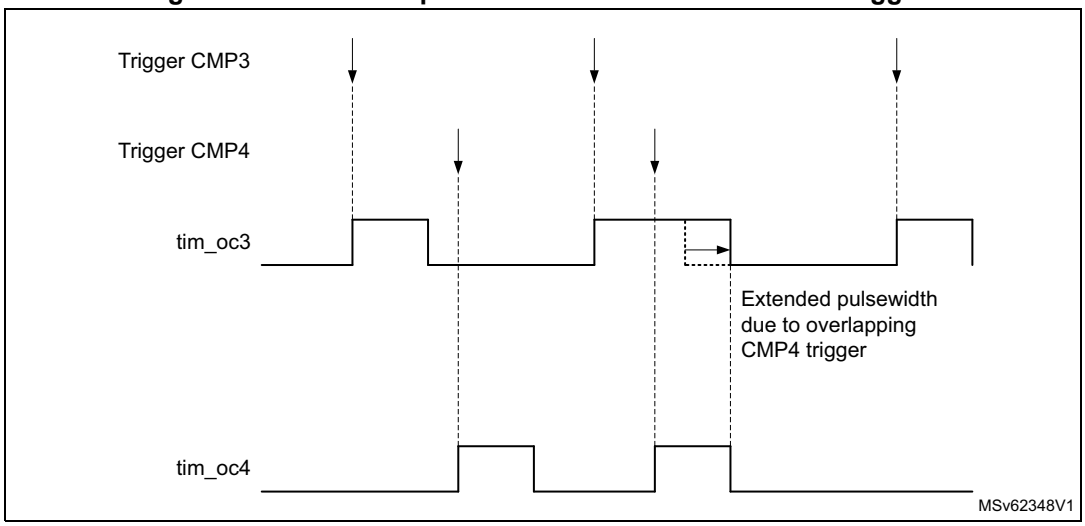
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 356).

Figure 356. Extended pulsewidth in case of concurrent triggers



35.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 357*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 357. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 357 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 357. Example of counter operation in encoder interface mode.

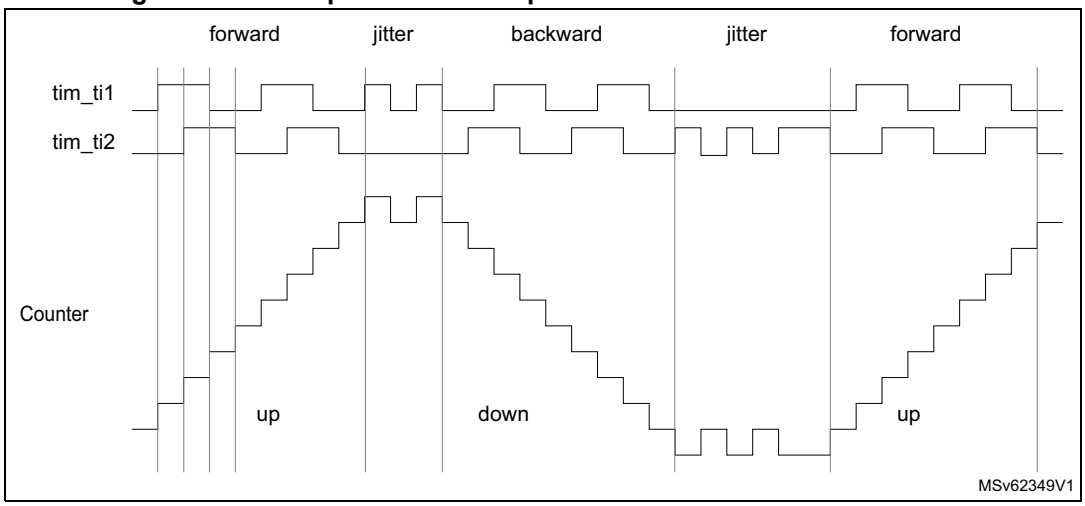


Figure 358 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 358. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

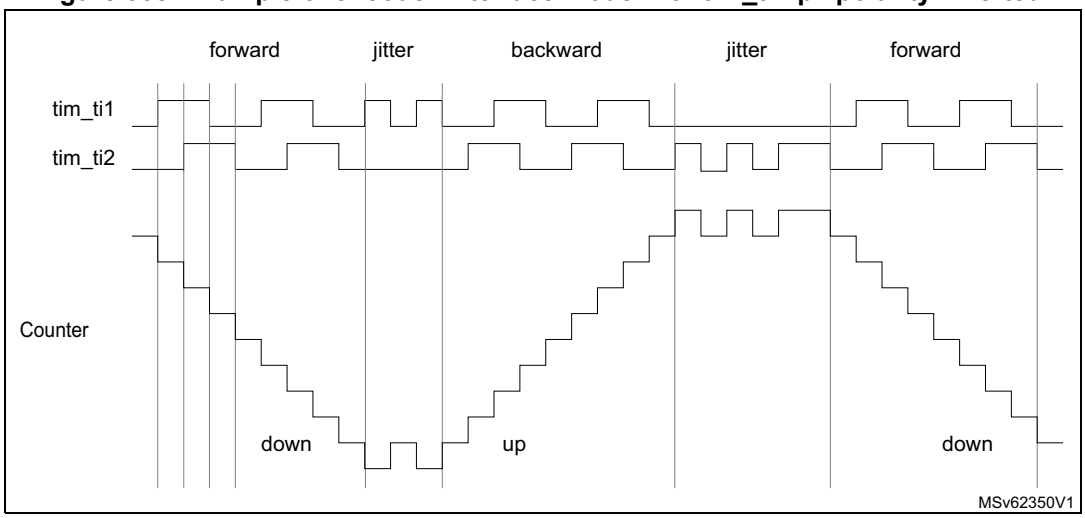
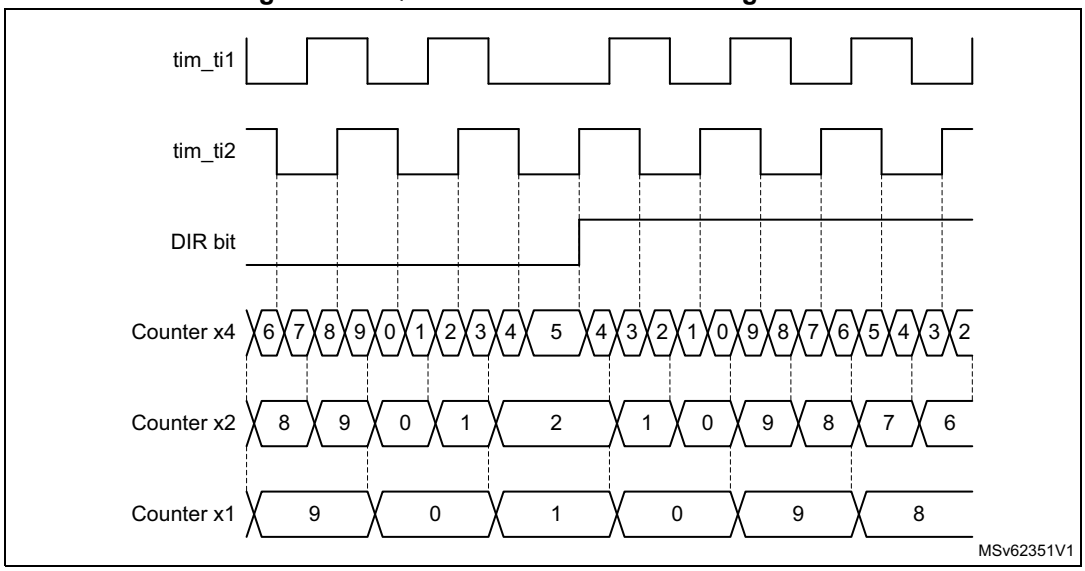


Figure 359 shows the timer counter value during a speed reversal, for various counting modes.

Figure 359. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

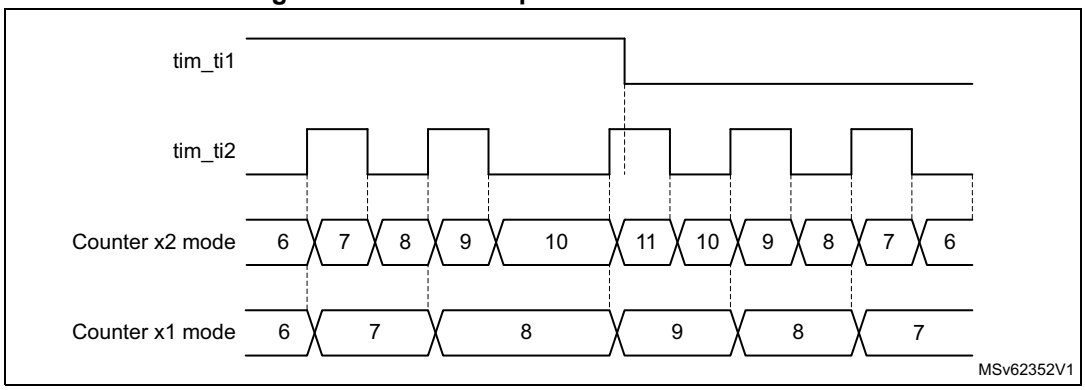
In the clock plus direction mode shown on *Figure 360*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 360. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 361*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 361*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 362*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 361*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 362*).

Figure 361. Directional clock encoder mode (CC1P = CC2P = 0)

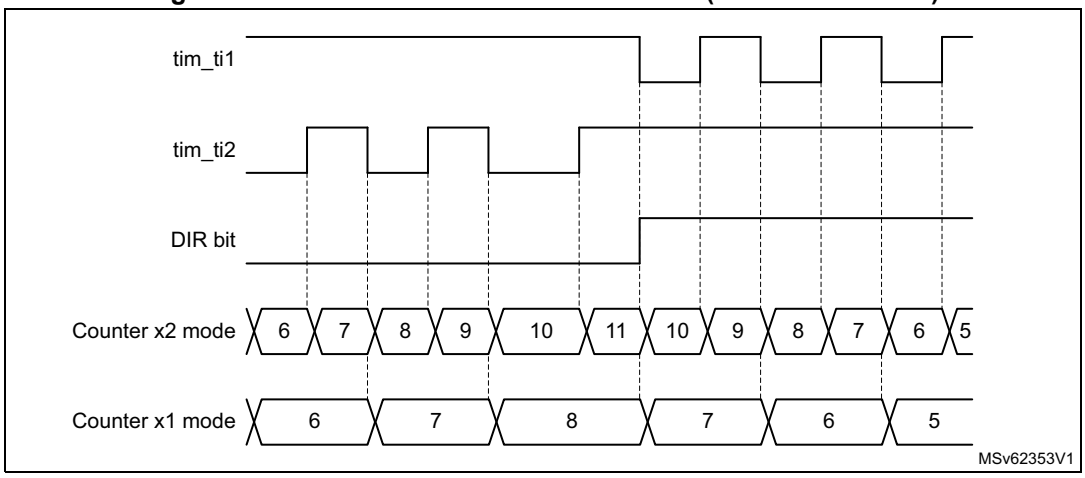


Figure 362. Directional clock encoder mode (CC1P = CC2P = 1)

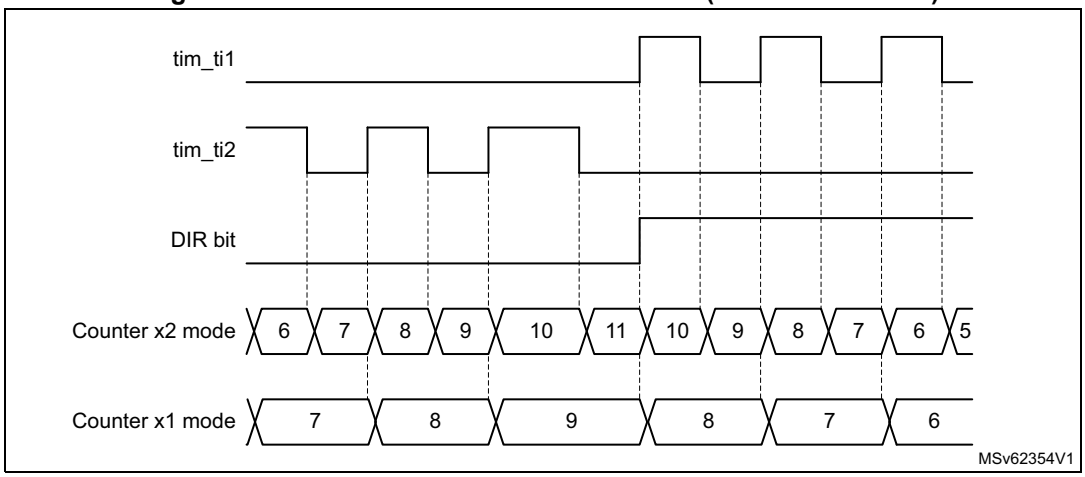


Table 358 here-below details how the directional clock mode operates, for any input transition.

Table 358. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

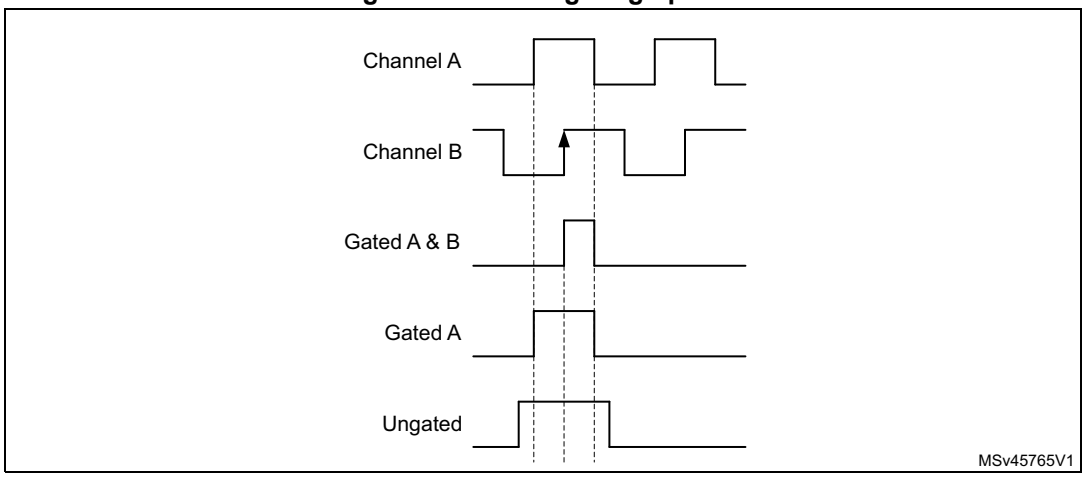
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 363*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

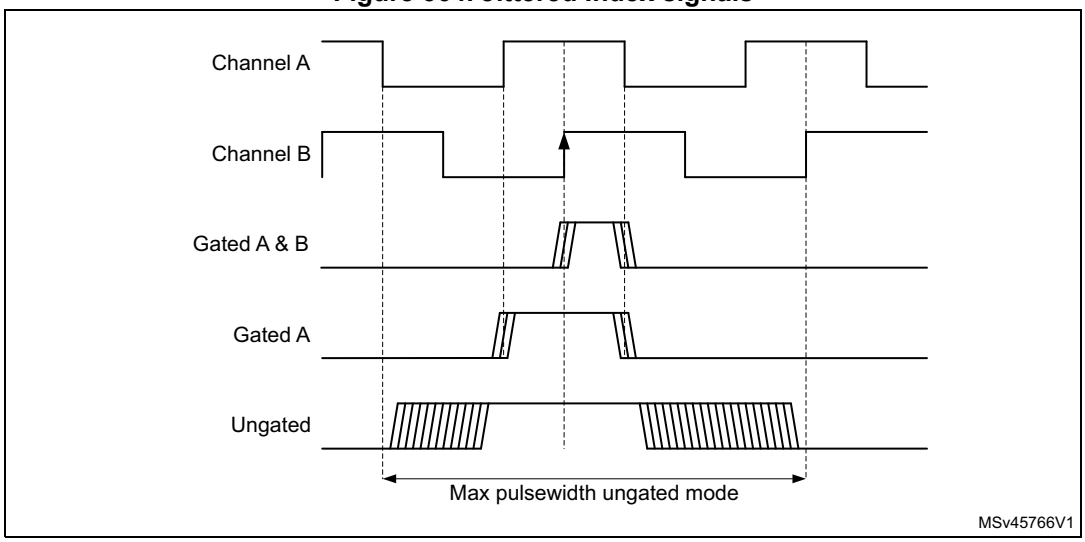
Figure 363. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 364*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 364. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 365* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 365. Index generation for IPOS[1:0] = 11

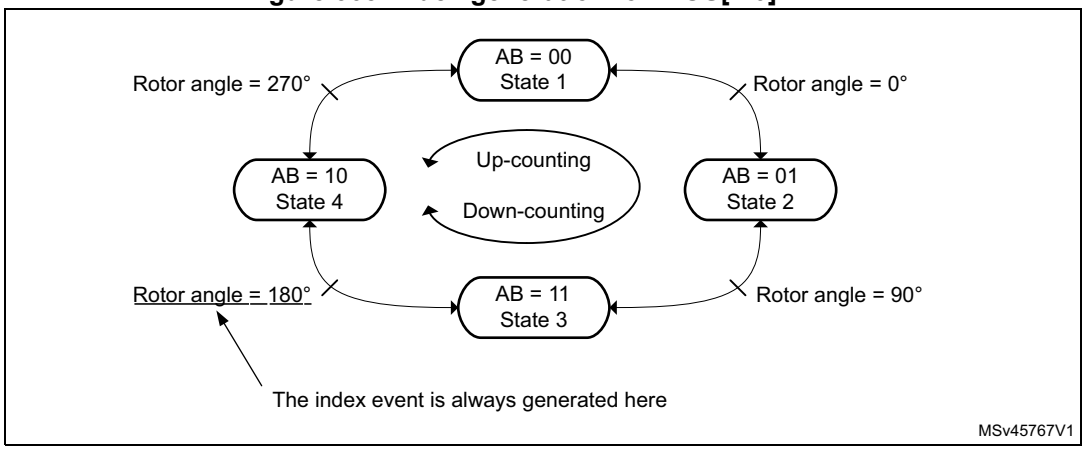


Figure 366 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 366. Counter reading with index gated on channel A (IPOS[1:0] = 11)

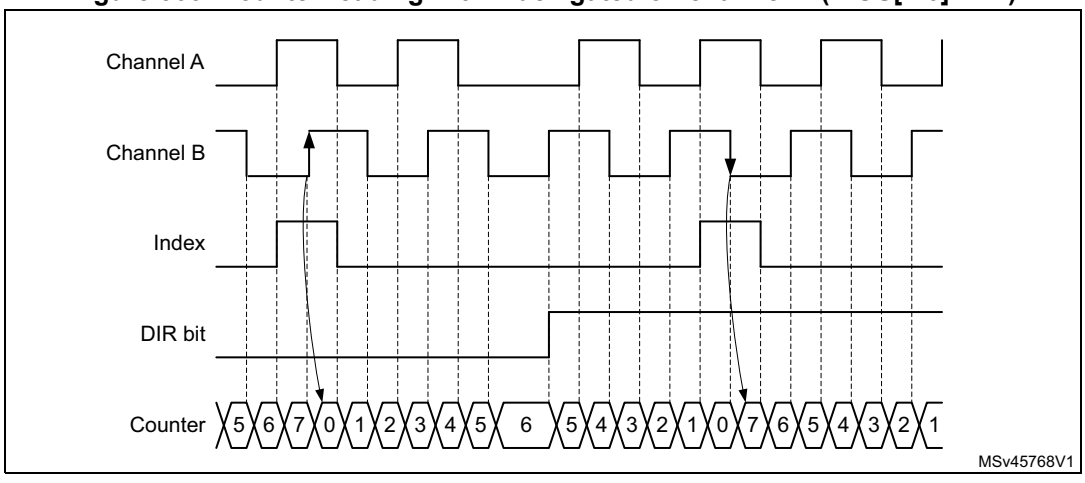


Figure 367. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 367. Counter reading with index ungated (IPOS[1:0] = 00)

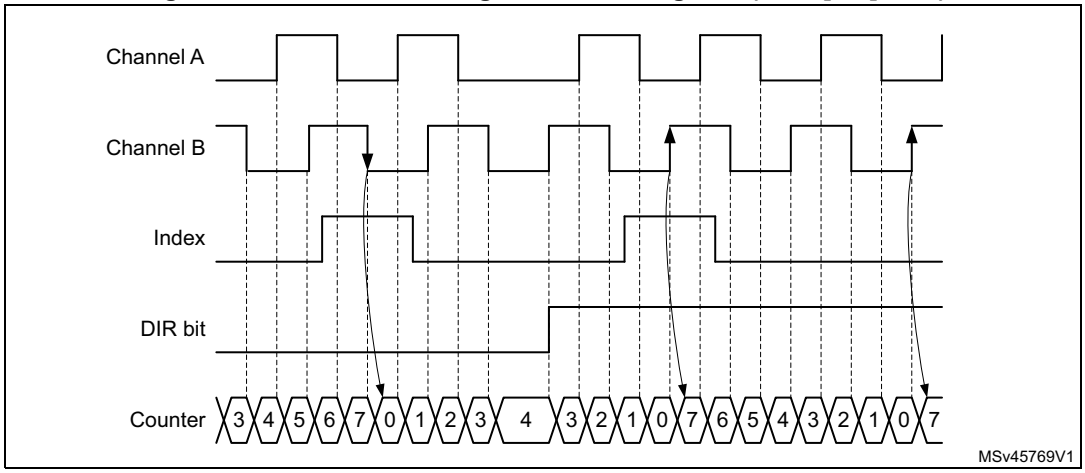


Figure 368. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 368. Counter reading with index gated on channel A and B

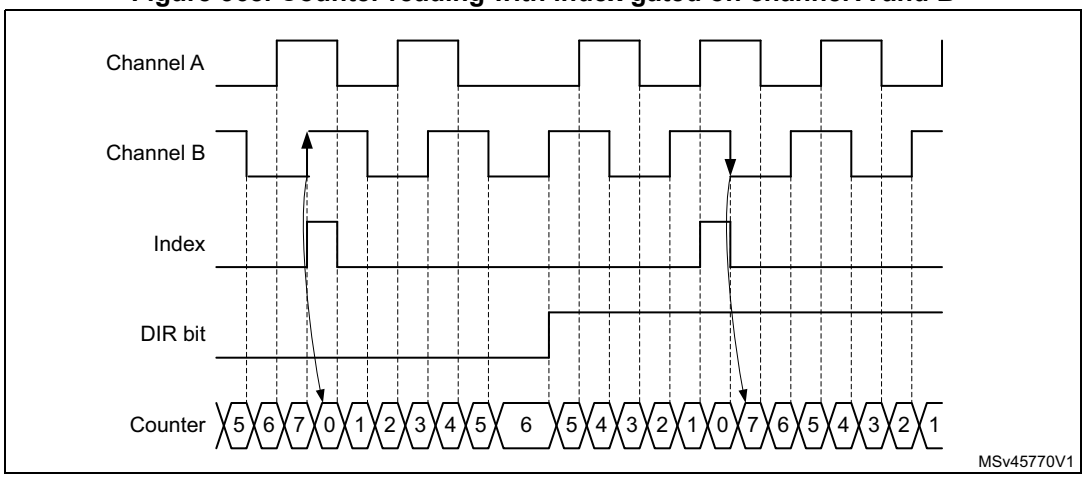


Figure 369 and *Figure 370* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 369. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

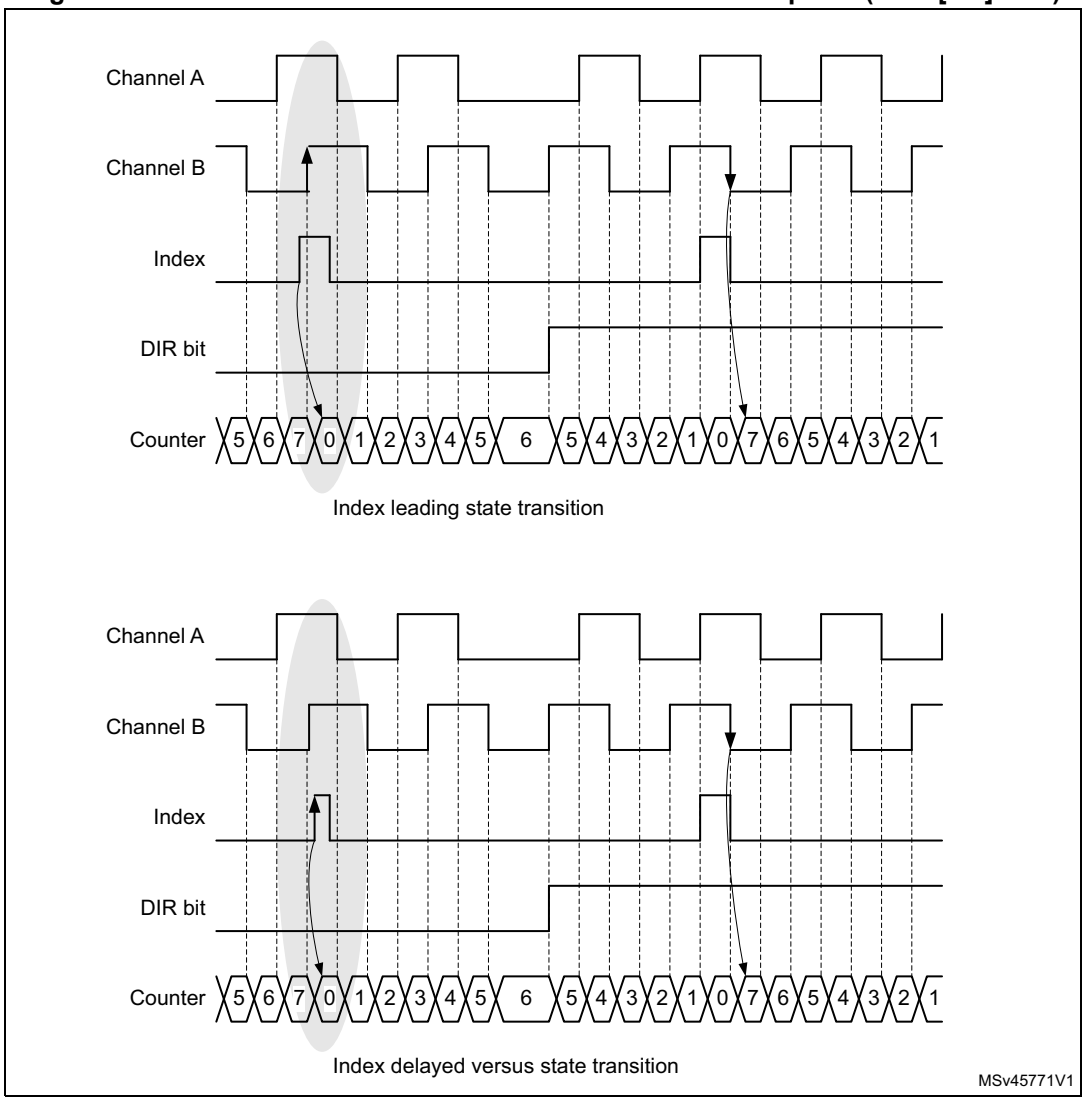


Figure 370. Counter reset Narrow index pulse (closer view, ARR = 0x07)

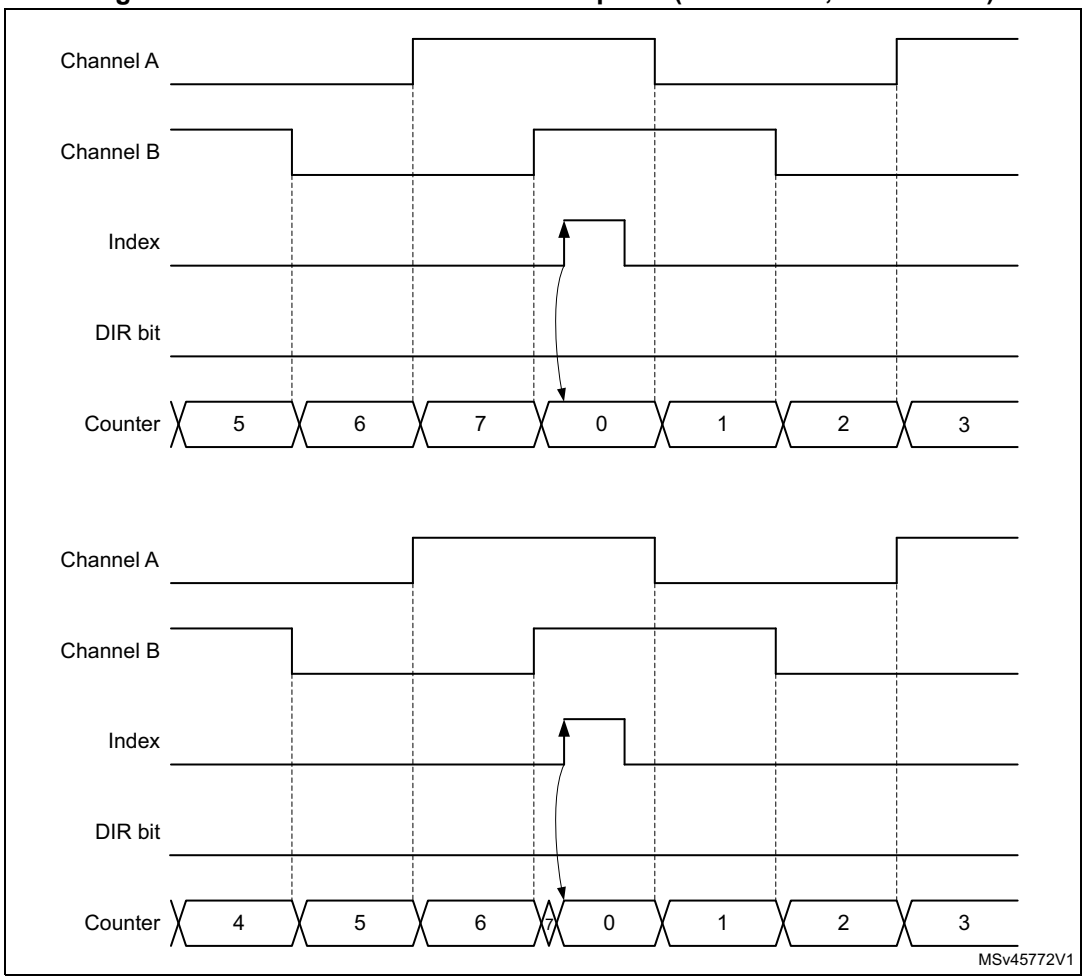
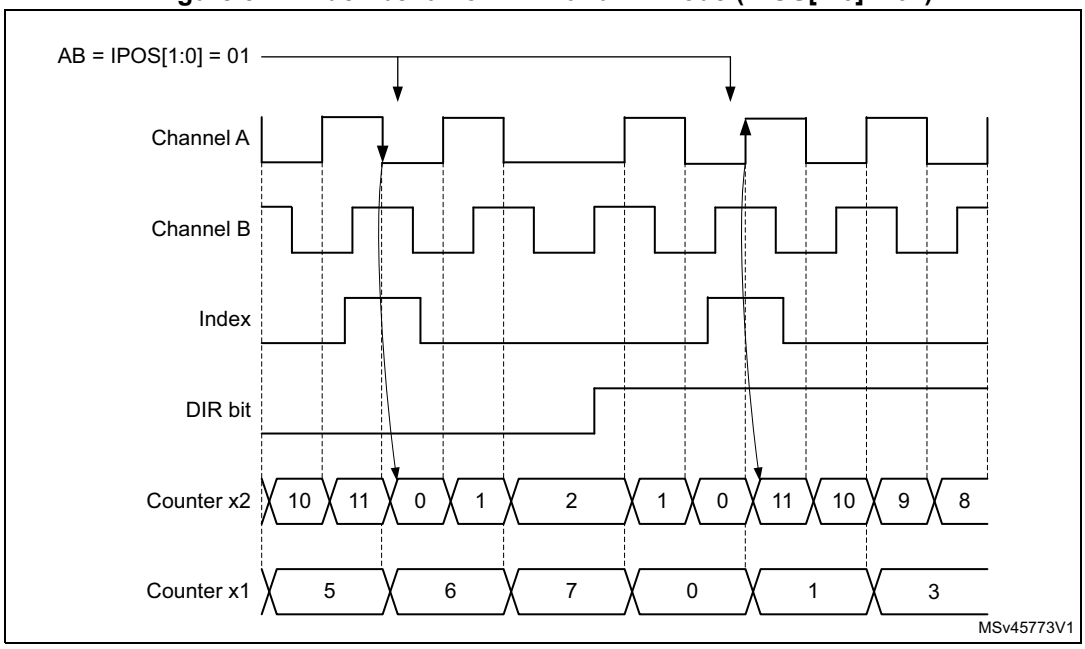


Figure 371 shows how the index is managed in x1 and x2 modes.

Figure 371. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

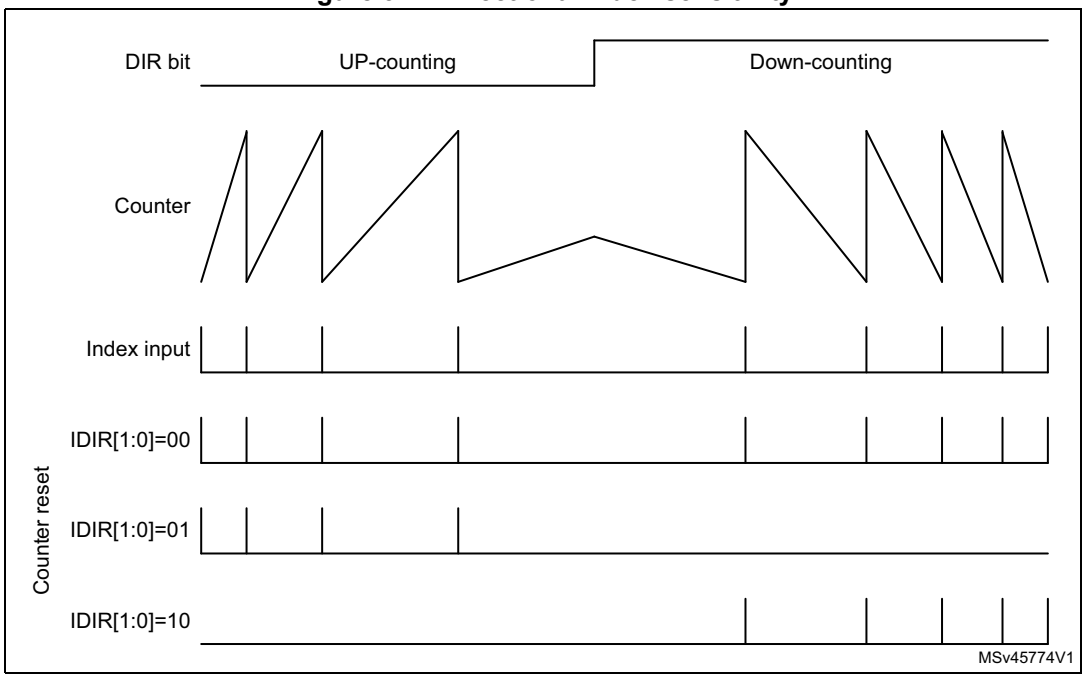


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 372 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

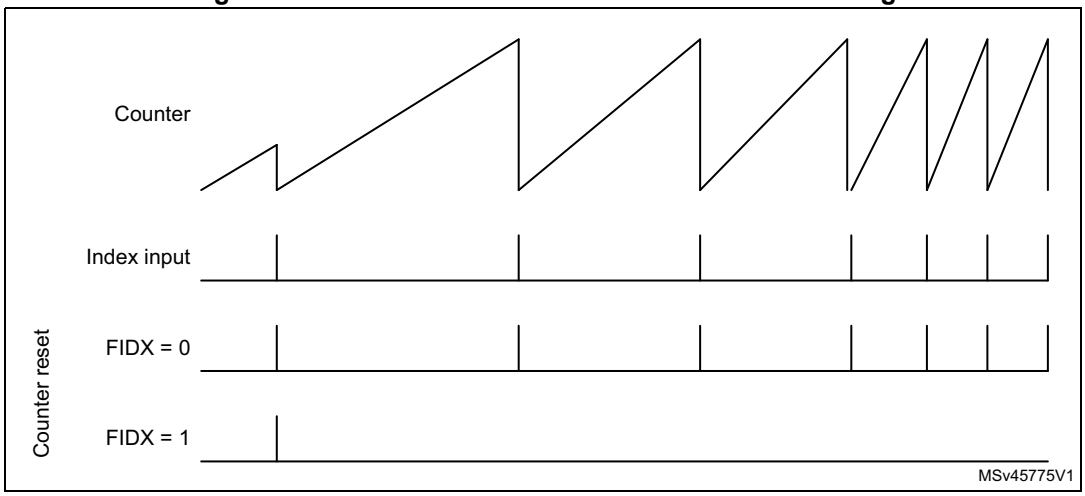
Figure 372. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 373*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 373. Counter reset as function of FIDX bit setting



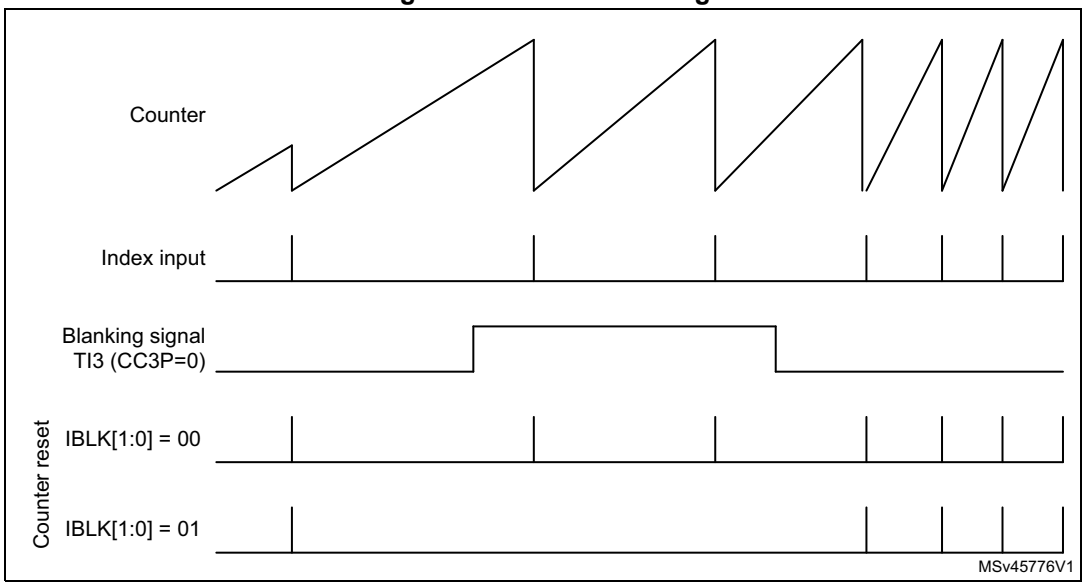
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 374* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 374. Index blanking



Index management in nonquadrature mode

Figure 375 and *Figure 376* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 375. Index behavior in clock + direction mode, IPOS[0] = 1

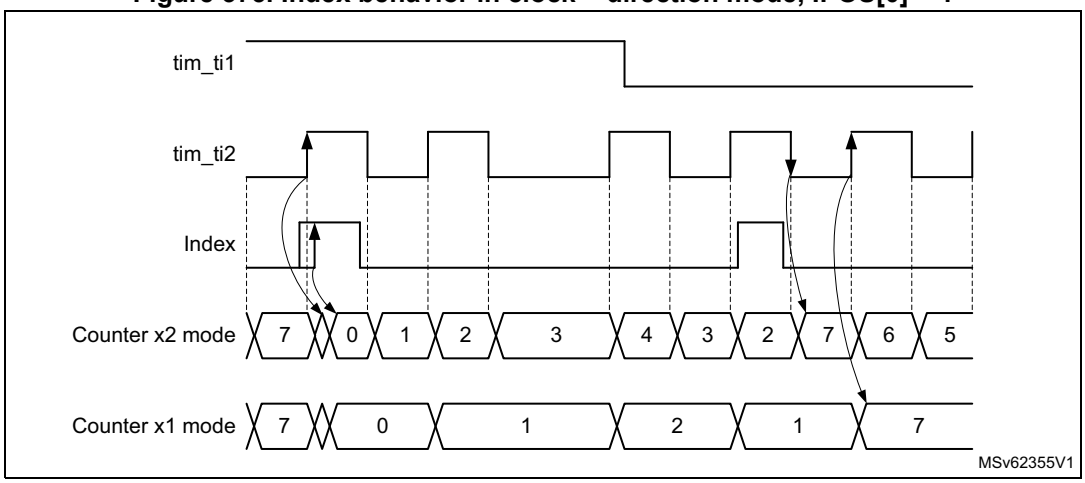
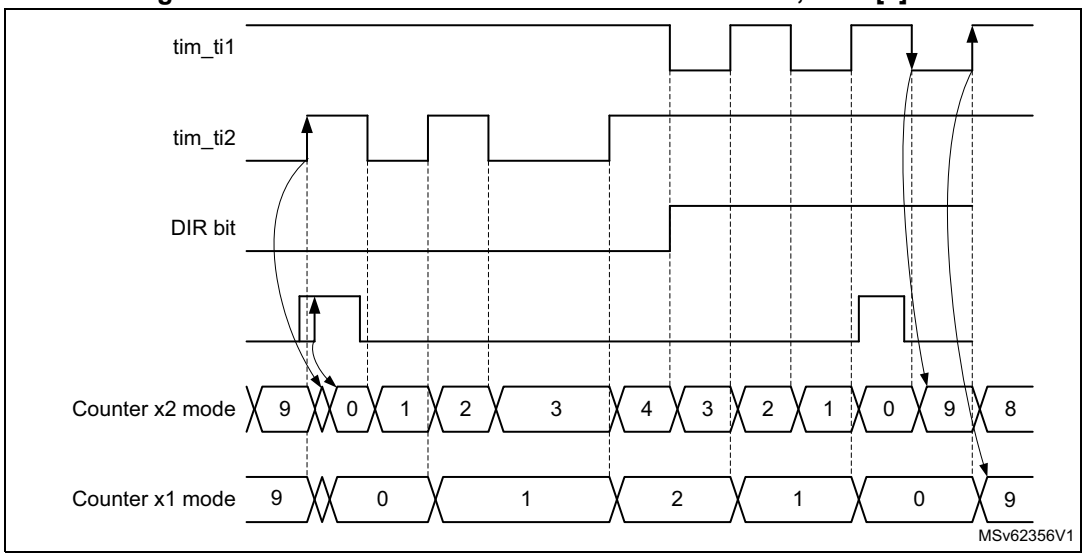


Figure 376. Index behavior in directional clock mode, IPOS[0] = 1

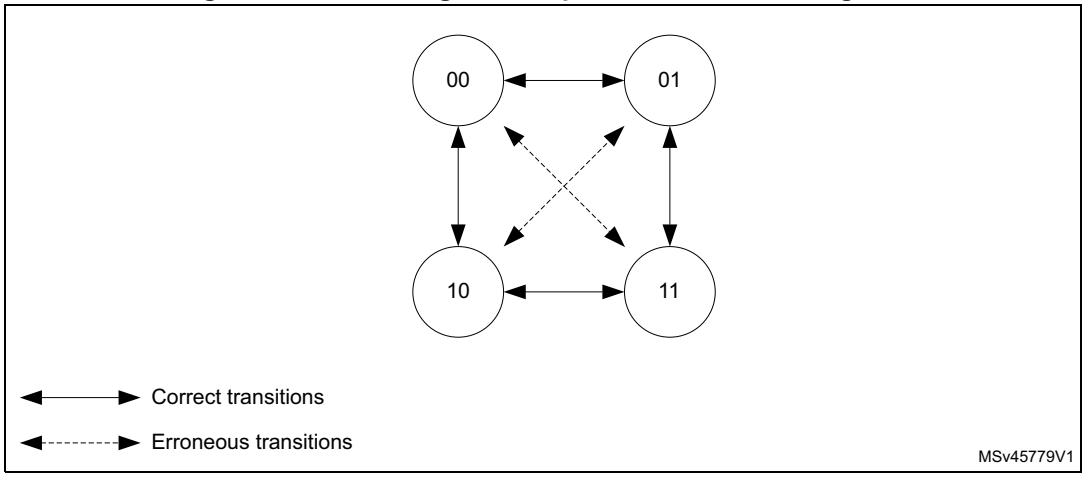


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 377.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 377. State diagram for quadrature encoded signals



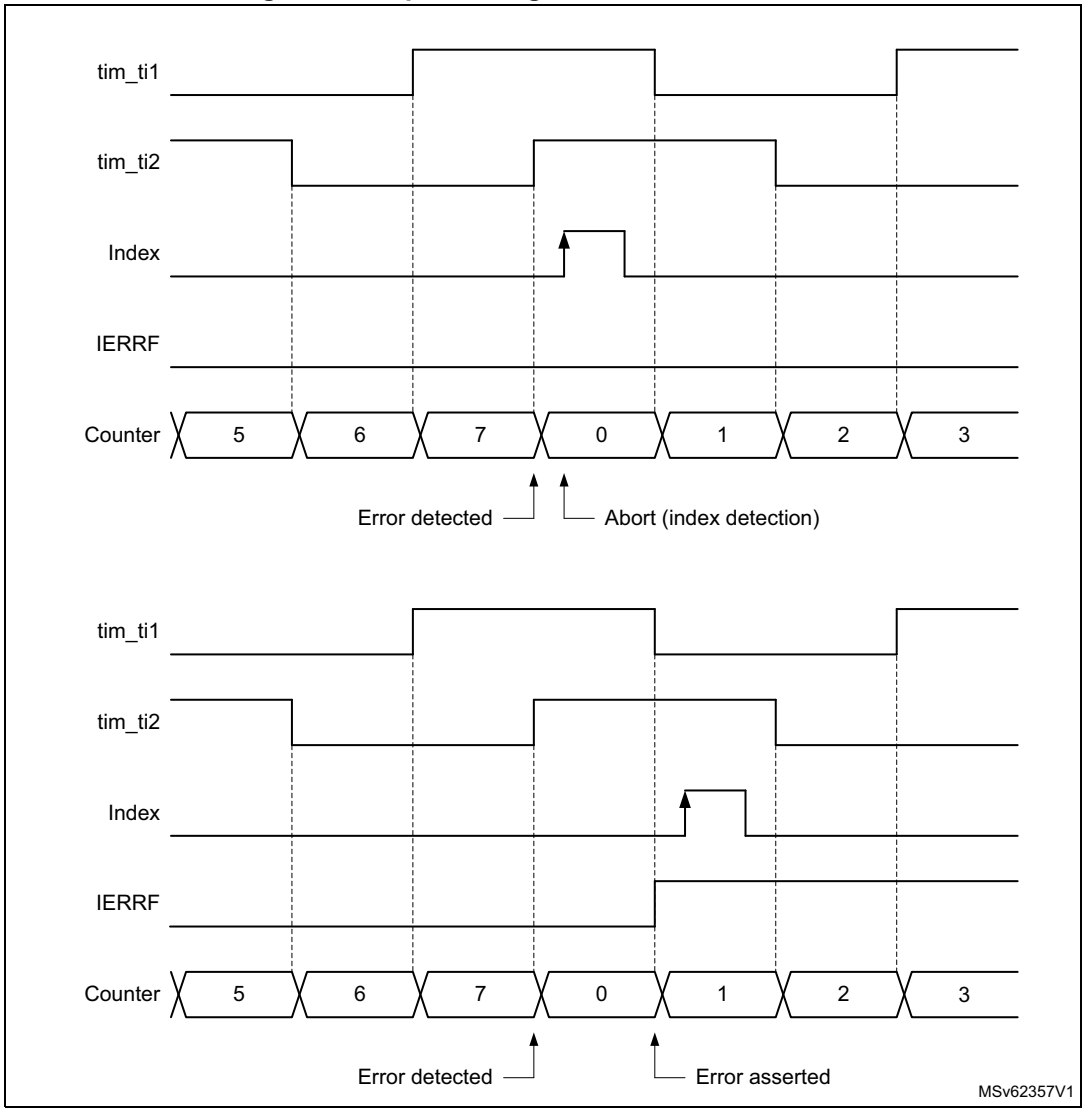
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

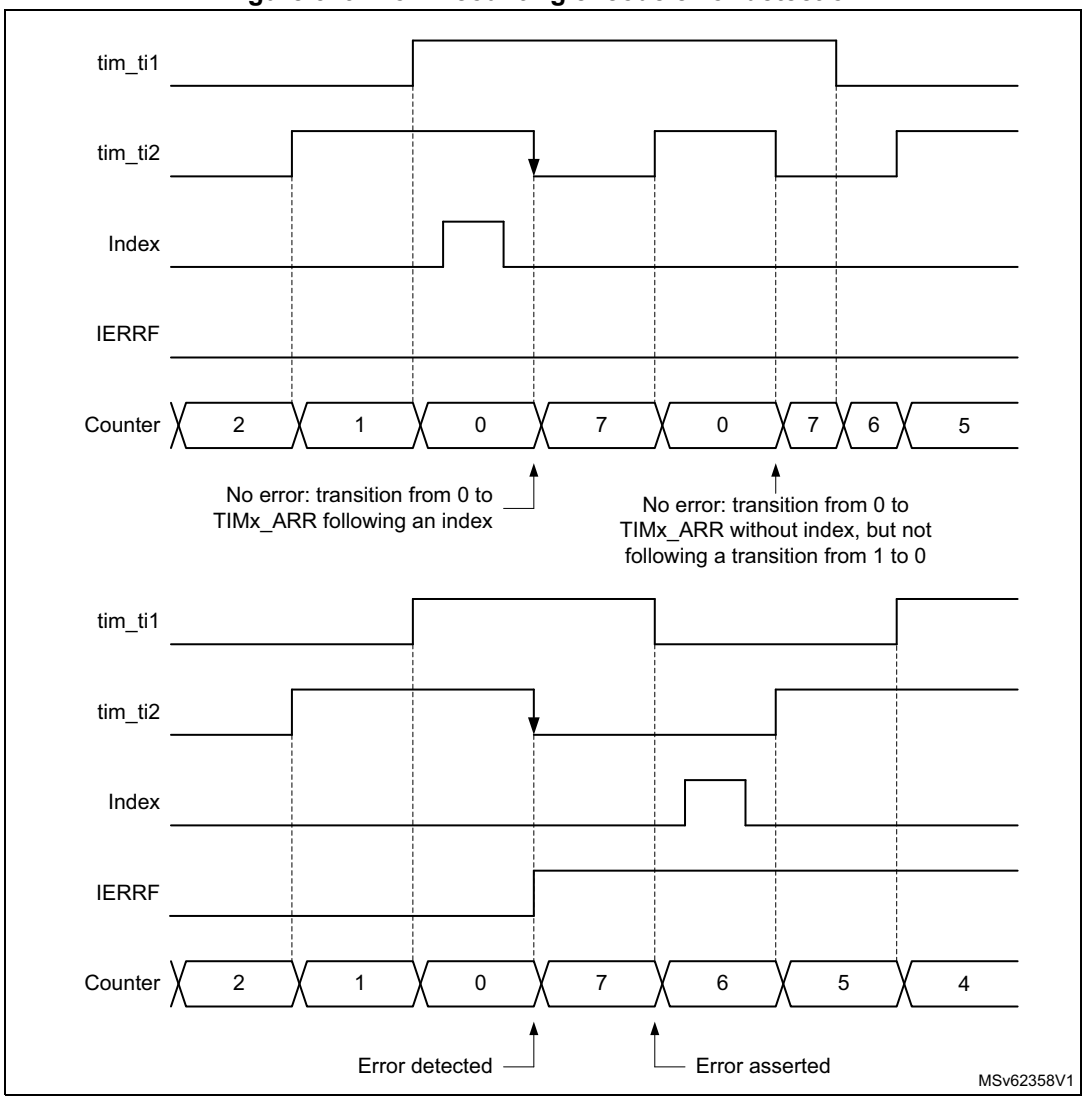
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 378*.

Figure 378. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 379*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 379. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

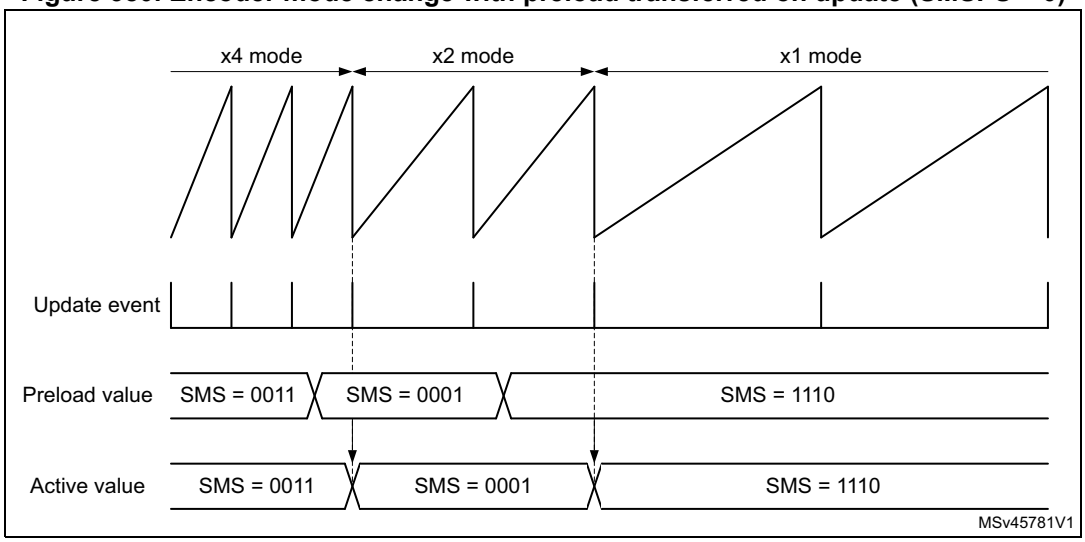
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 380*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 380. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

35.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

35.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

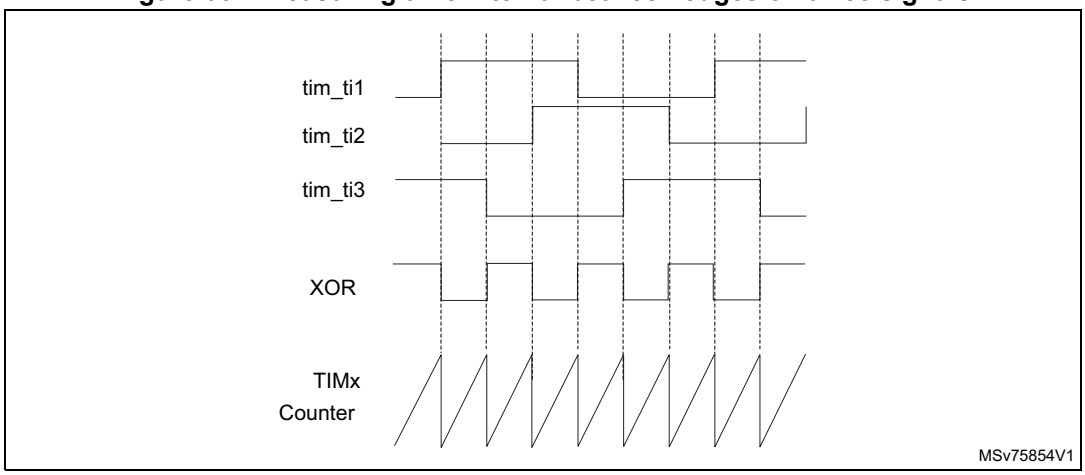
There is no latency between the UIF and UIFCPY flags assertion.

35.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 381*.

Figure 381. Measuring time interval between edges on three signals



35.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 382*. The "interfacing timer" captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 324*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

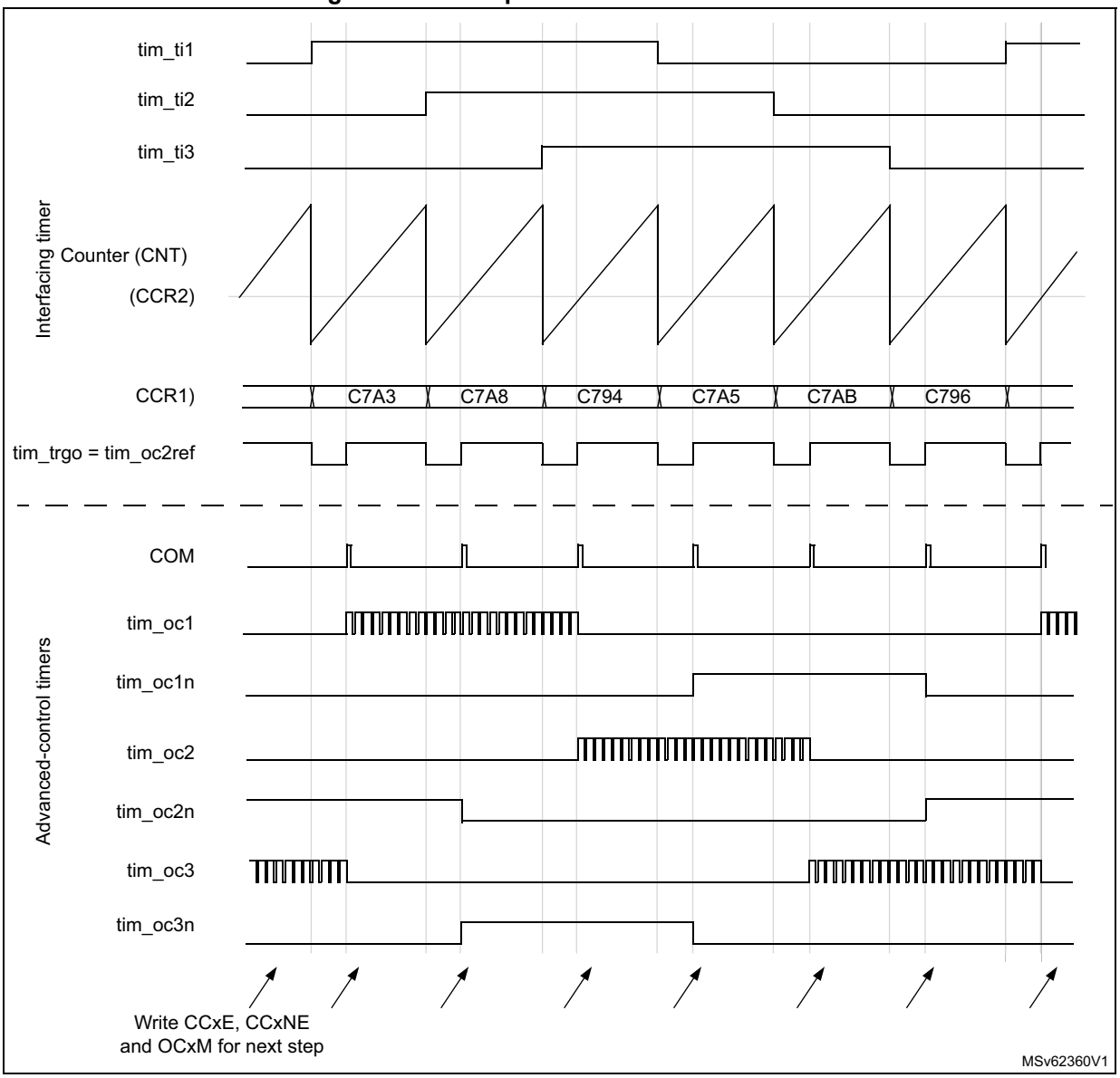
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 382 describes this example.

Figure 382. Example of Hall sensor interface



35.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 36.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

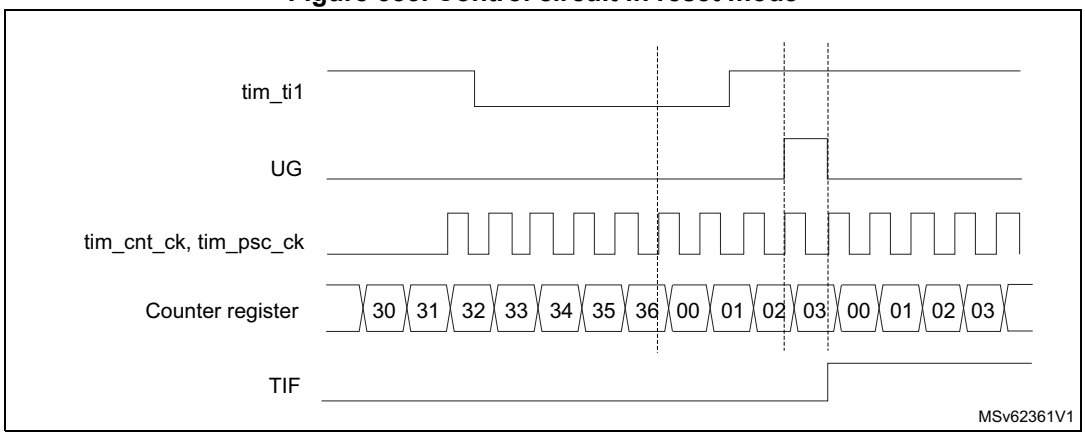
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 383. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

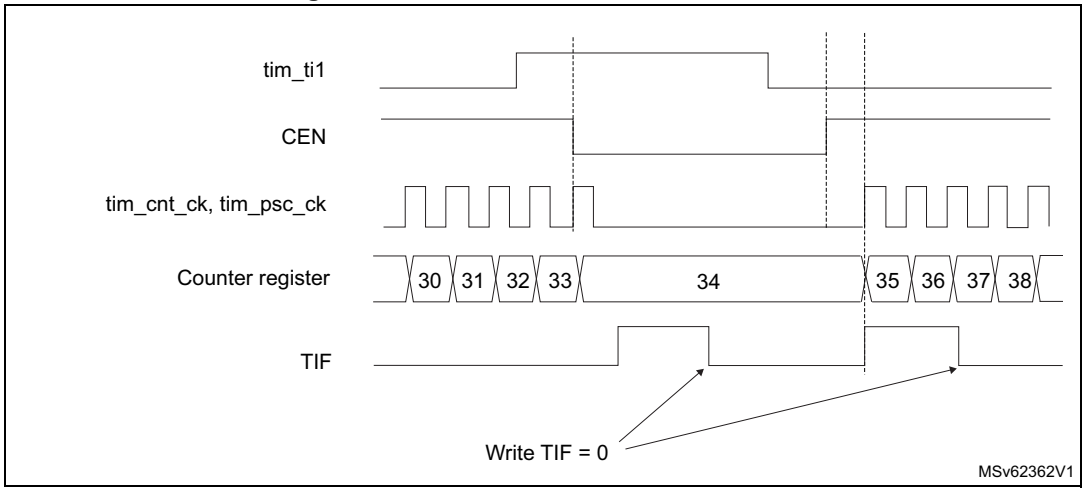
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 384. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

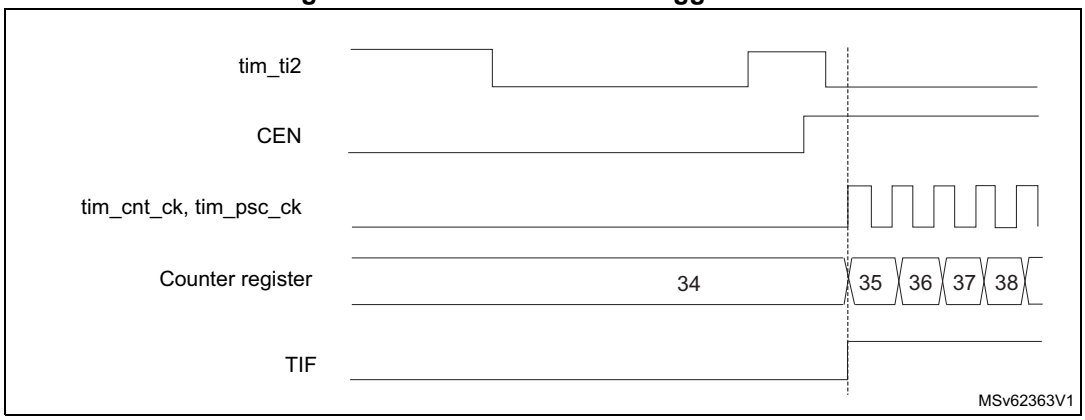
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 385. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

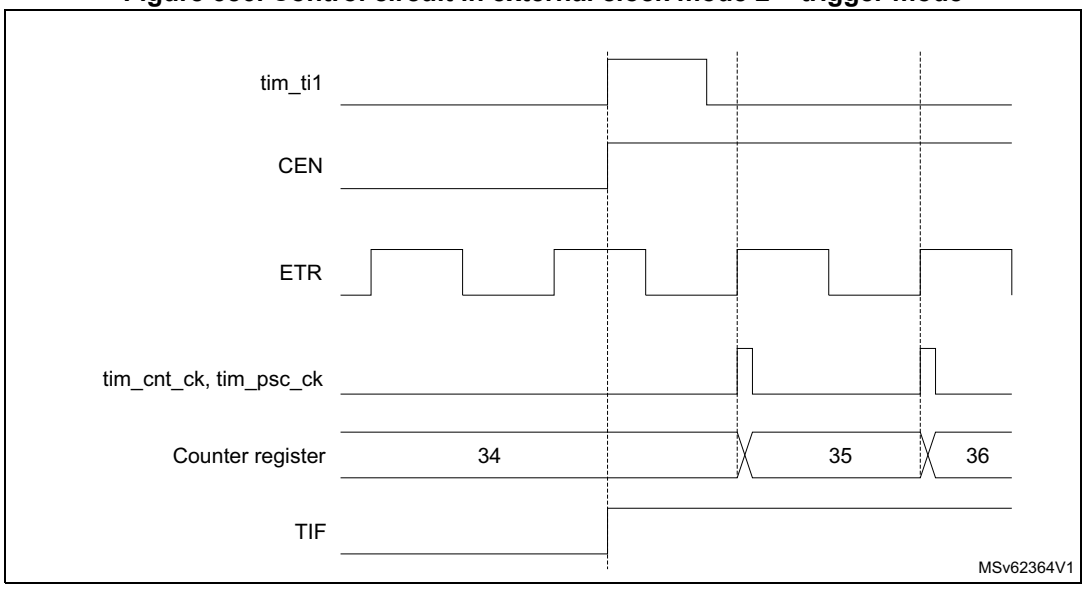
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 386. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 339*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

35.3.32 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period:

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period:

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

35.3.33 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 35.6.29: TIM1 DMA control register (TIM1_DCR)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address.
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (see note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

35.3.34 TIM1 DMA requests

The TIM1 can generate a DMA request, as shown in the table below.

Table 359. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

35.3.35 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

35.4 TIM1 low-power modes

Table 360. Effect of low-power modes on TIM1

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

35.5 TIM1 interrupts

The TIM1 can generate multiple interrupts, as shown in *Table 361*.

Table 361. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

35.6 TIM1 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

35.6.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

35.6.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
			rw			rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 35.3.32* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

35.6.3 TIM1 slave mode control register (TIM1_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Table 347: Internal trigger connection* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

35.6.4 TIM1 DMA/interrupt enable register (TIM1_DIER)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

35.6.5 TIM1 status register (TIM1_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 35.6.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

35.6.6 TIM1 event generation register (TIM1_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

35.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

35.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

35.6.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

35.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

35.6.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 362* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 362. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0, tim_ocxn = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref x or CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z).</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered).
Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration).
Note: tim_brk2 can only be used if OSSI = OSSR = 1.</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

35.6.12 TIM1 counter (TIM1_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

35.6.13 TIM1 prescaler (TIM1_PSC)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

35.6.14 TIM1 autoreload register (TIM1_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 35.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

35.6.15 TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

35.6.16 TIM1 capture/compare register 1 (TIM1_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

35.6.17 TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

35.6.18 TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

35.6.19 TIM1 capture/compare register 4 (TIM1_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

35.6.20 TIM1 break and dead-time register (TIM1_BDTR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode

1: Break input tim_brk in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed

1: Break input tim_brk is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

0: Break input tim_brk2 is active low

1: Break input tim_brk2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection, see *Figure 344: Break and Break2 circuitry overview*).

0: Break2 function disabled

1: Break2 function enabled

Note: The BRKIN2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 35.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 344: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 35.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 35.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.21 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

35.6.22 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

35.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

35.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

35.6.25 TIM1 timer encoder control register (TIM1_ECR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input
00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

35.6.26 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 35.3.2: TIM1 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 35.3.2: TIM1 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 35.3.2: TIM1 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 35.3.2: TIM1 pins and internal signals* for interconnects list.

35.6.27 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 35.3.2: TIM1 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 35.3.2: TIM1 pins and internal signals for product specific implementation.

35.6.28 TIM1 alternate function register 2 (TIM1_AF2)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 35.3.2: TIM1 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 35.3.2: TIM1 pins and internal signals for product specific implementation.

35.6.29 TIM1 DMA control register (TIM1_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

35.6.30 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

35.6.31 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 363. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value				0			0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]			MSM	TS[2:0]			OCCS	SMS[2:0]			
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 363. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIMx_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 363. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	TIMx_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIMx_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIMx_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIMx_AF2	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 General-purpose timers (TIM2/TIM3/TIM4)

36.1 TIM2/TIM3/TIM4 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 36.4.23: Timer synchronization*.

36.2 TIM2/TIM3/TIM4 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

36.3 TIM2/TIM3/TIM4 implementation

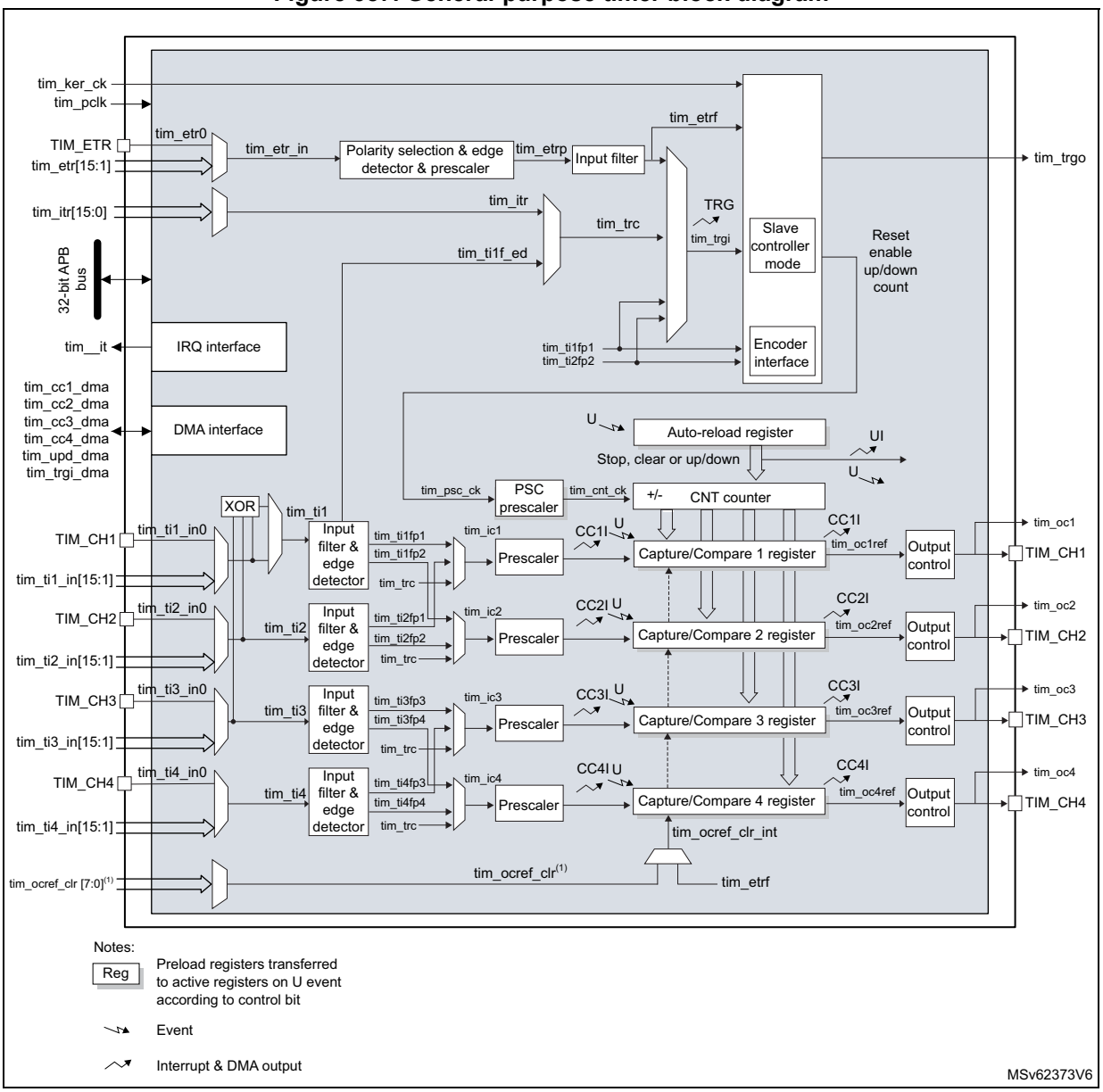
Table 364. STM32U3 series general purpose timers

Timer instance	TIM2	TIM3	TIM4
Resolution	32-bit	32-bit	32-bit
OCREF clear selection Sources	Yes tim_etrf tim_ocref_clr[7:0]	Yes tim_etrf tim_ocref_clr[7:0]	Yes tim_etrf tim_ocref_clr[7:0]

36.4 TIM2/TIM3/TIM4 functional description

36.4.1 Block diagram

Figure 387. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 36.3: TIM2/TIM3/TIM4 implementation*.

36.4.2 TIM2/TIM3/TIM4 pins and internal signals

Table 365 and *Table 366* in this section summarize the TIM inputs and outputs.

Table 365. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 366. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 366. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 367. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM2	TIM3	TIM4
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out
tim_ti1_in2	comp2_out	comp2_out	comp2_out
tim_ti1_in[15:3]	Reserved		

Table 368. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources		
	TIM2	TIM3	TIM4
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2
tim_ti2_in1	comp1_out	comp1_out	comp1_out
tim_ti2_in2	comp2_out	comp2_out	comp2_out
tim_ti2_in[15:3]	Reserved		

Table 369. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources		
	TIM2	TIM3	TIM4
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3
tim_ti3_in[15:1]	Reserved		

Table 370. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources		
	TIM2	TIM3	TIM4
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4
tim_ti4_in1	comp1_out	Reserved	
tim_ti4_in2	comp2_out		
tim_ti4_in[15:3]	Reserved		

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 371. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved
tim_itr4	Reserved		
tim_itr5			
tim_itr6	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr7	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr[15:9]	Reserved		

Tables below list the internal sources connected to the tim_etr input multiplexer.

Table 372. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment		
	TIM2	TIM3	TIM4
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR
tim_etr1	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out
tim_etr3	MSIK	MSIK	MSIK
tim_etr4	HSI	HSI	HSI
tim_etr5	Reserved		MSIS
tim_etr6	Reserved		
tim_etr7			
tim_etr8	TIM3_ETR	TIM2_ETR	TIM3_ETR

Table 372. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signal assignment		
	TIM2	TIM3	TIM4
tim_etr9	TIM4_ETR	TIM4_ETR	Reserved
tim_etr10	Reserved		
tim_etr11	LSE	adc1_awd1	
tim_etr12	Reserved	adc1_awd2	
tim_etr13	USB_SOF	adc1_awd3	
tim_etr[15:14]	Reserved		

The table below lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 373. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment		
	TIM2	TIM3	TIM4
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr[7:2]	Reserved		

36.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 388 and *Figure 389* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 388. Counter timing diagram with prescaler division change from 1 to 2

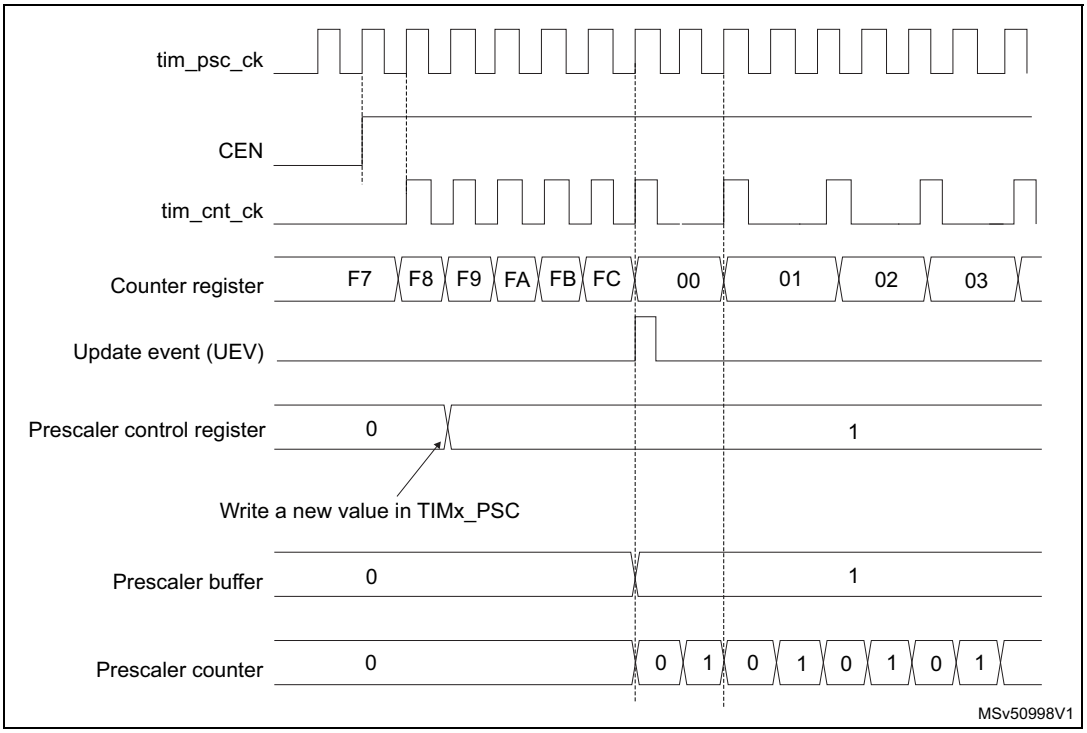
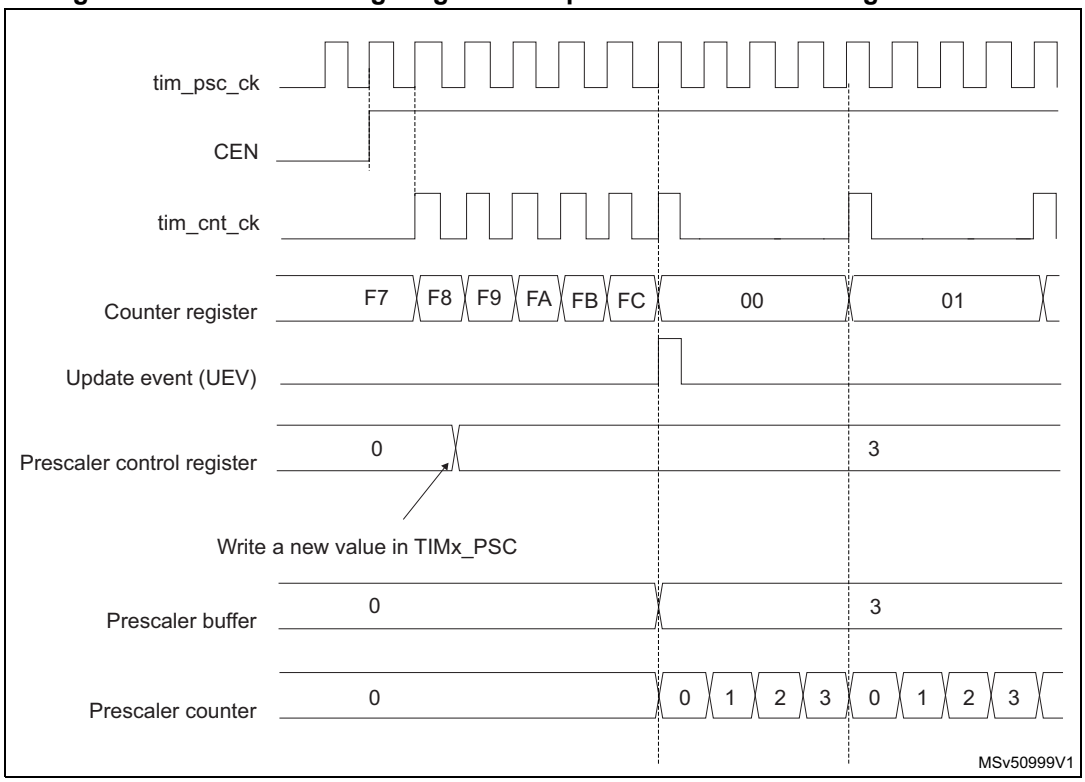


Figure 389. Counter timing diagram with prescaler division change from 1 to 4



36.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 390. Counter timing diagram, internal clock divided by 1

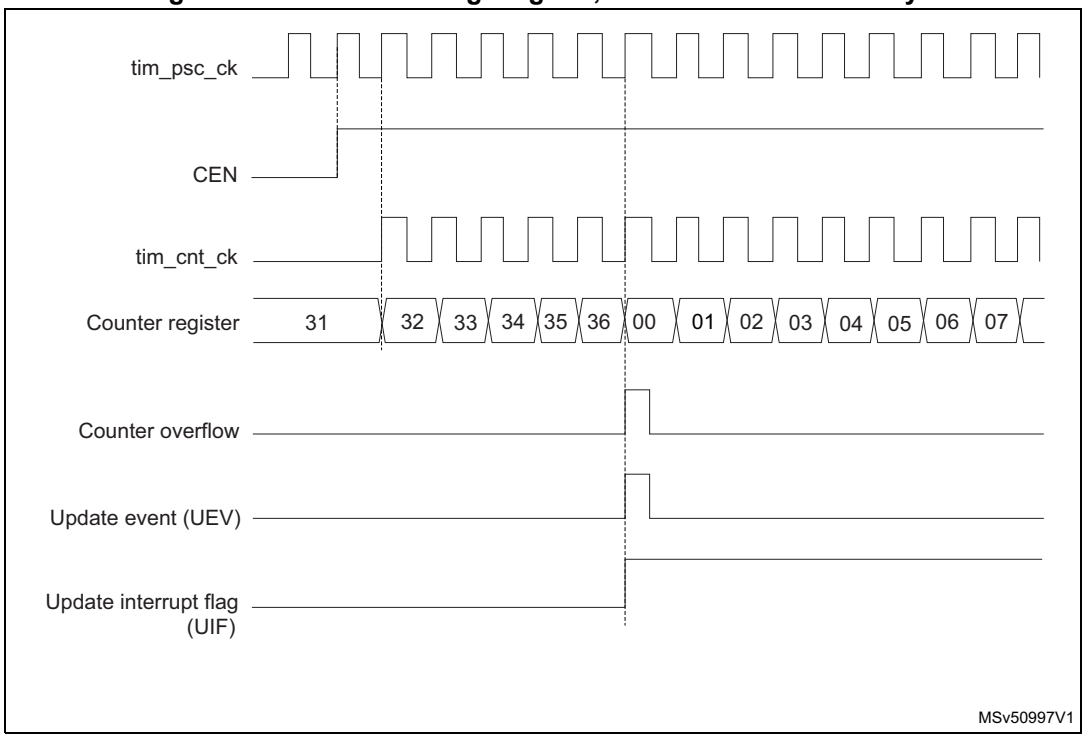


Figure 391. Counter timing diagram, internal clock divided by 2

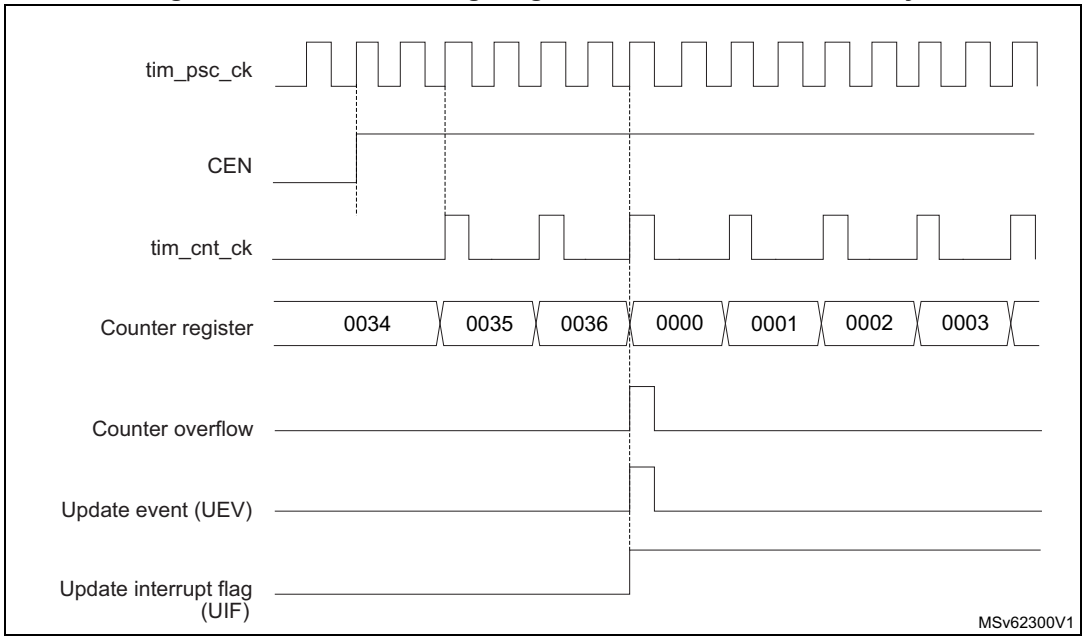


Figure 392. Counter timing diagram, internal clock divided by 4

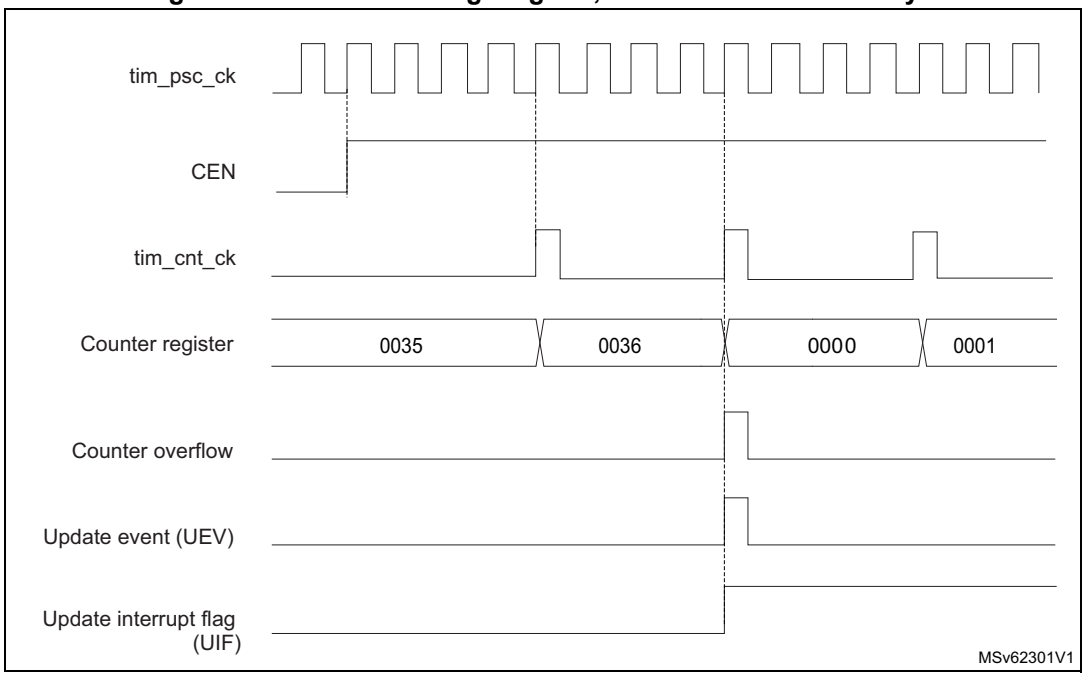


Figure 393. Counter timing diagram, internal clock divided by N

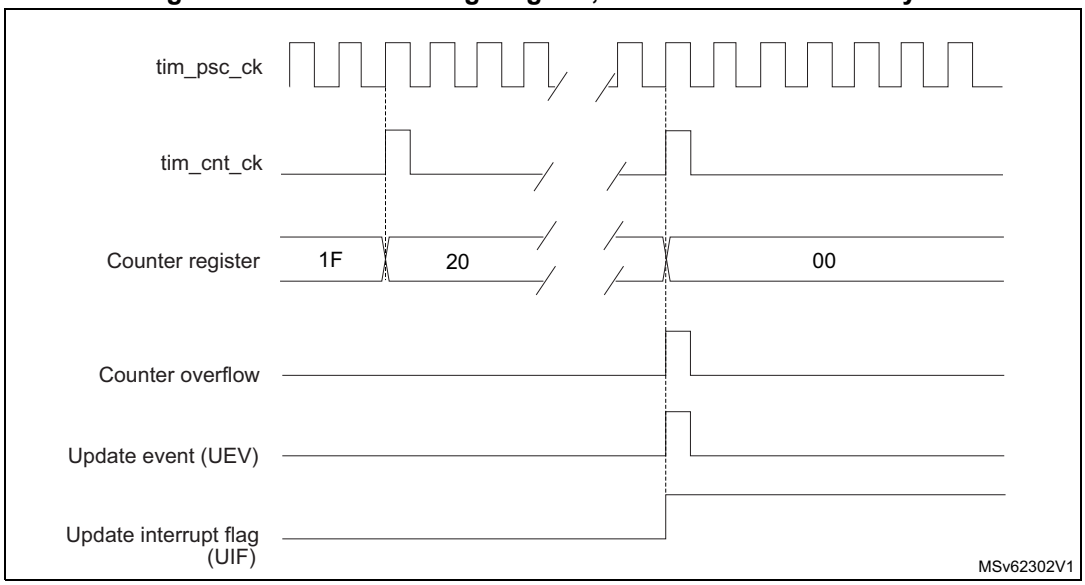


Figure 394. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

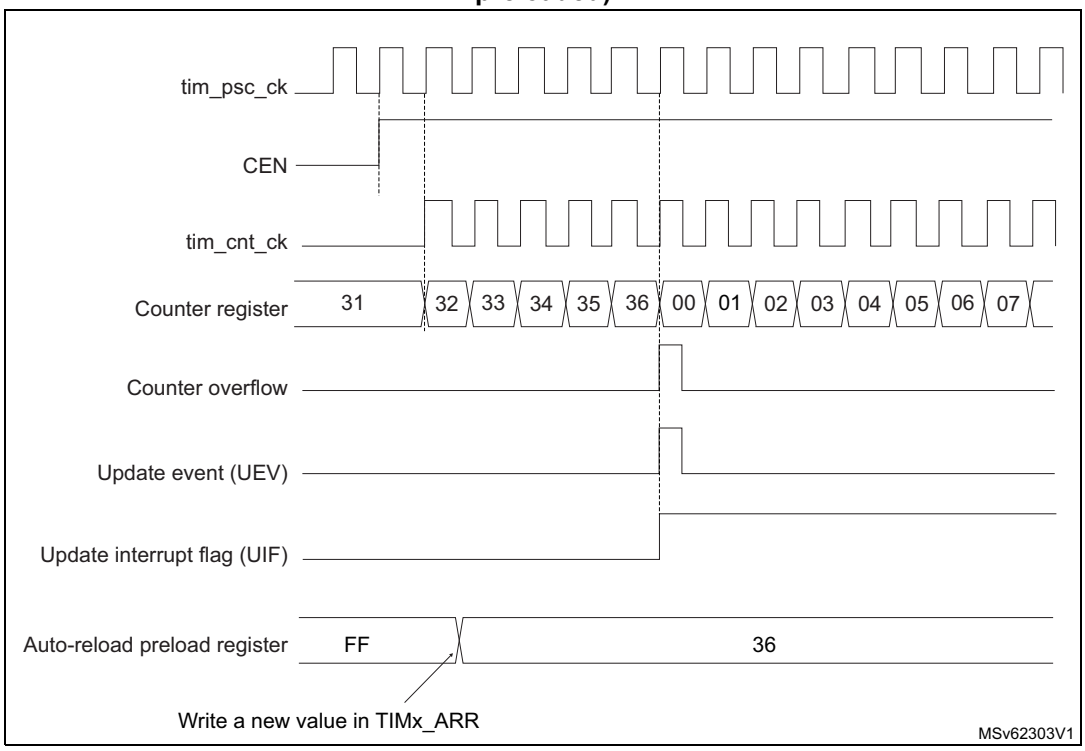
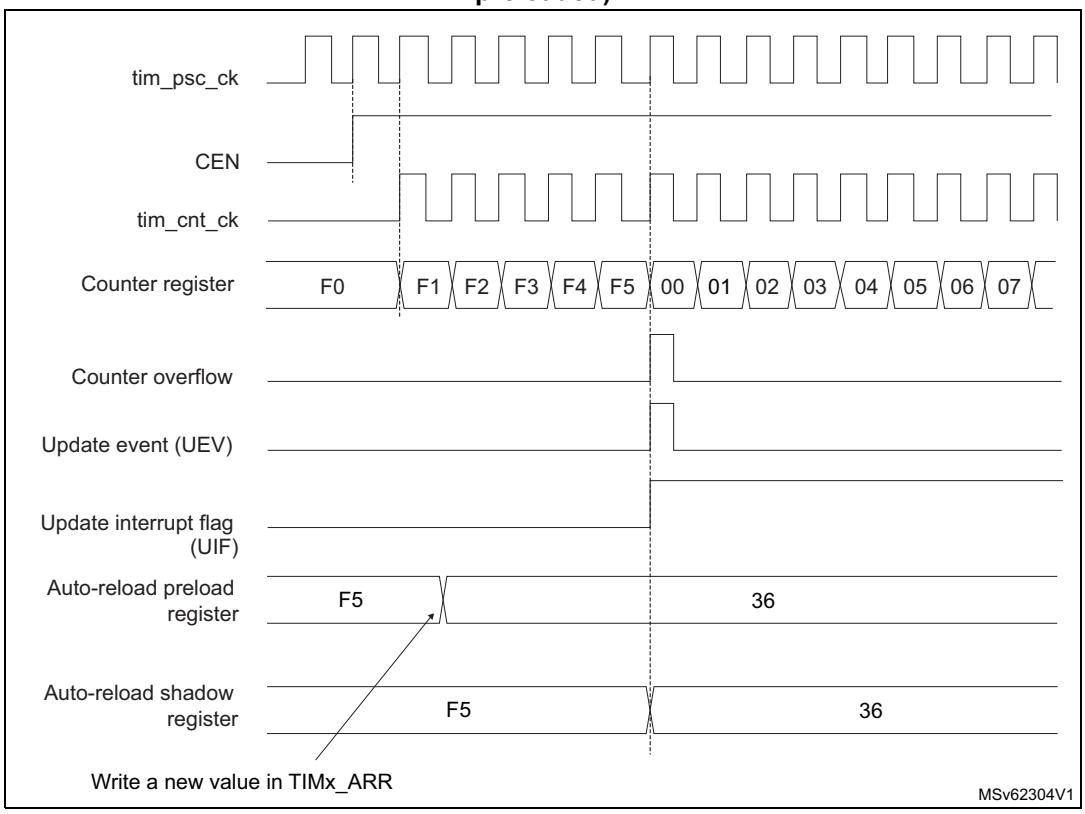


Figure 395. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 396. Counter timing diagram, internal clock divided by 1

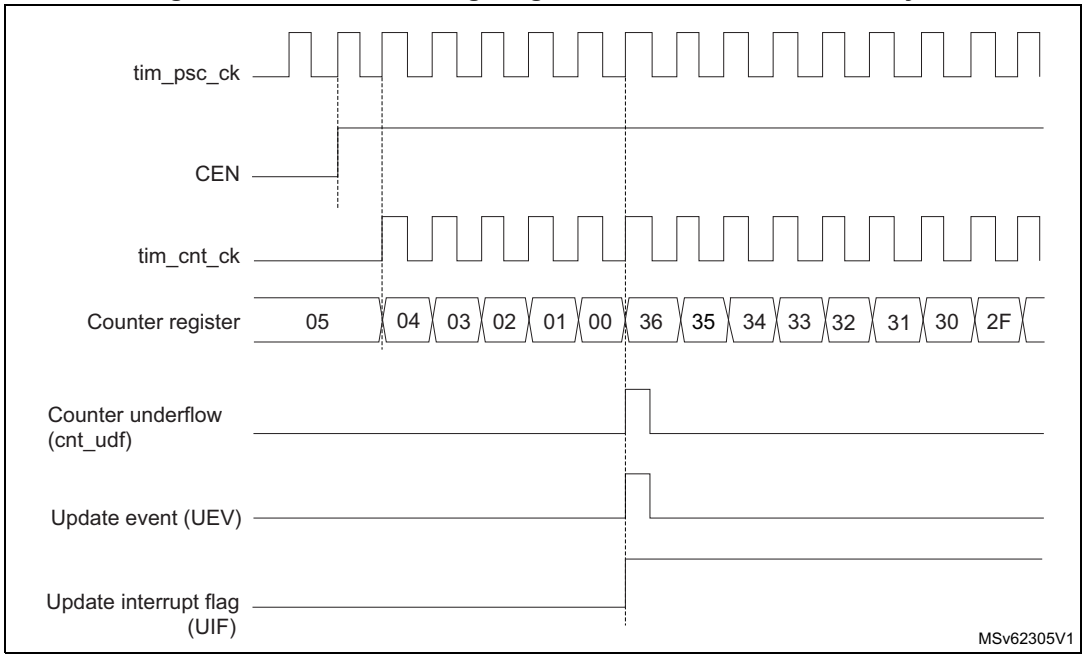


Figure 397. Counter timing diagram, internal clock divided by 2

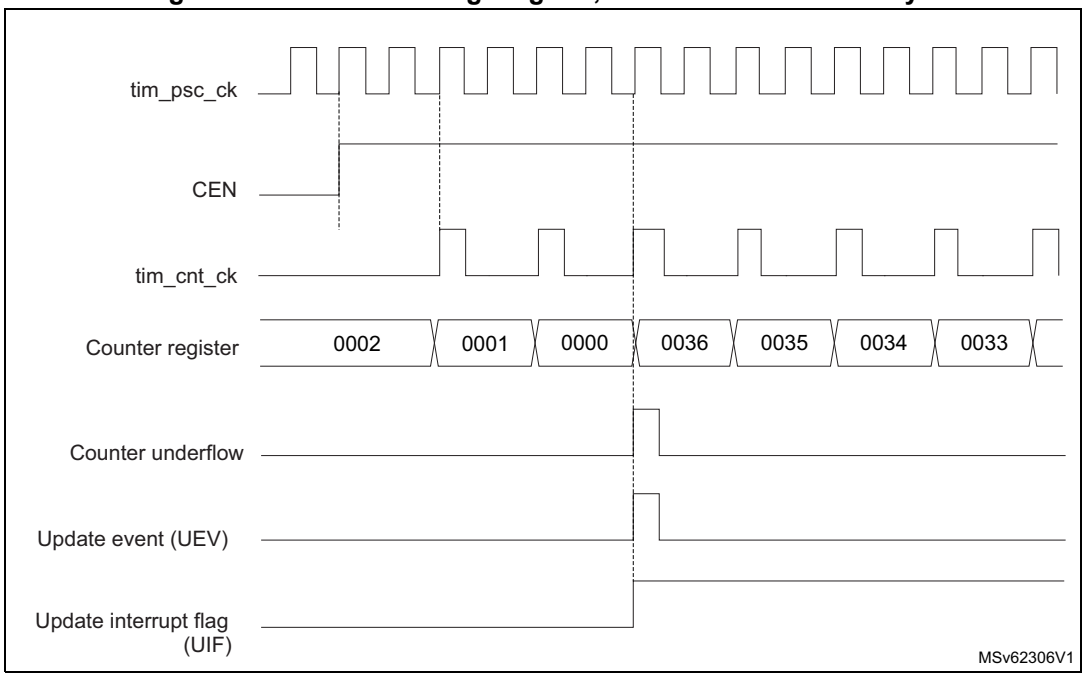


Figure 398. Counter timing diagram, internal clock divided by 4

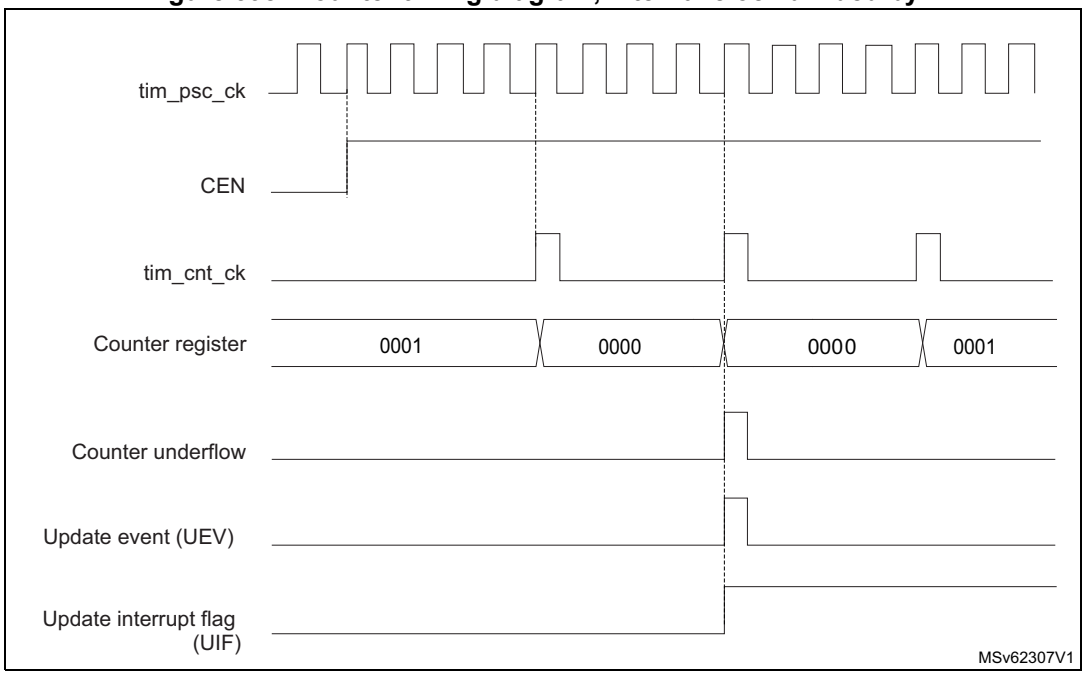


Figure 399. Counter timing diagram, internal clock divided by N

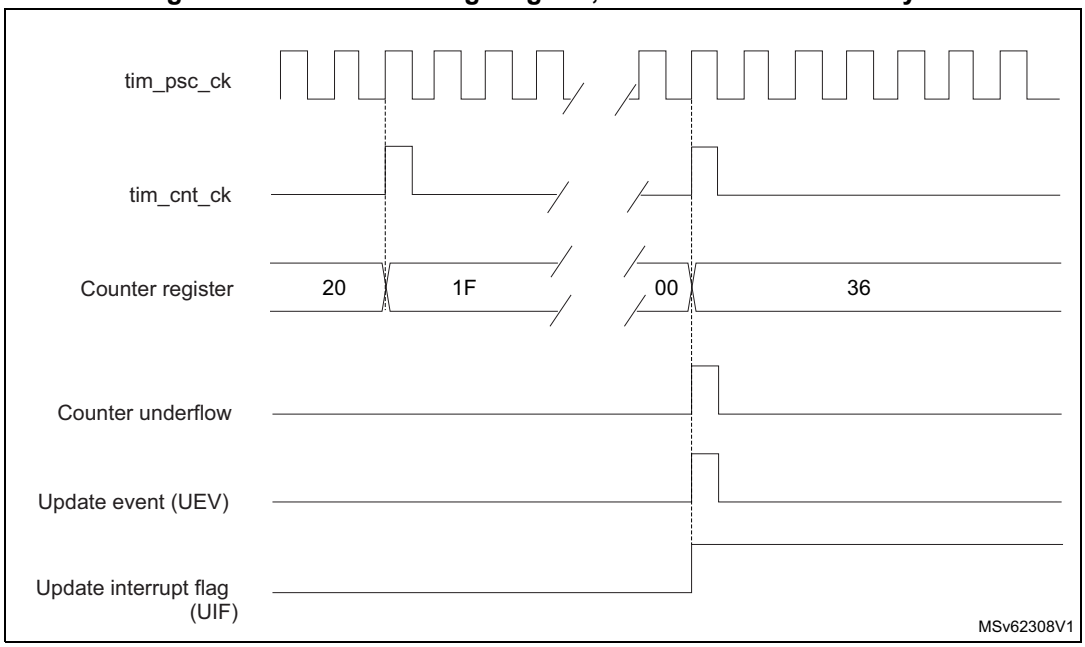
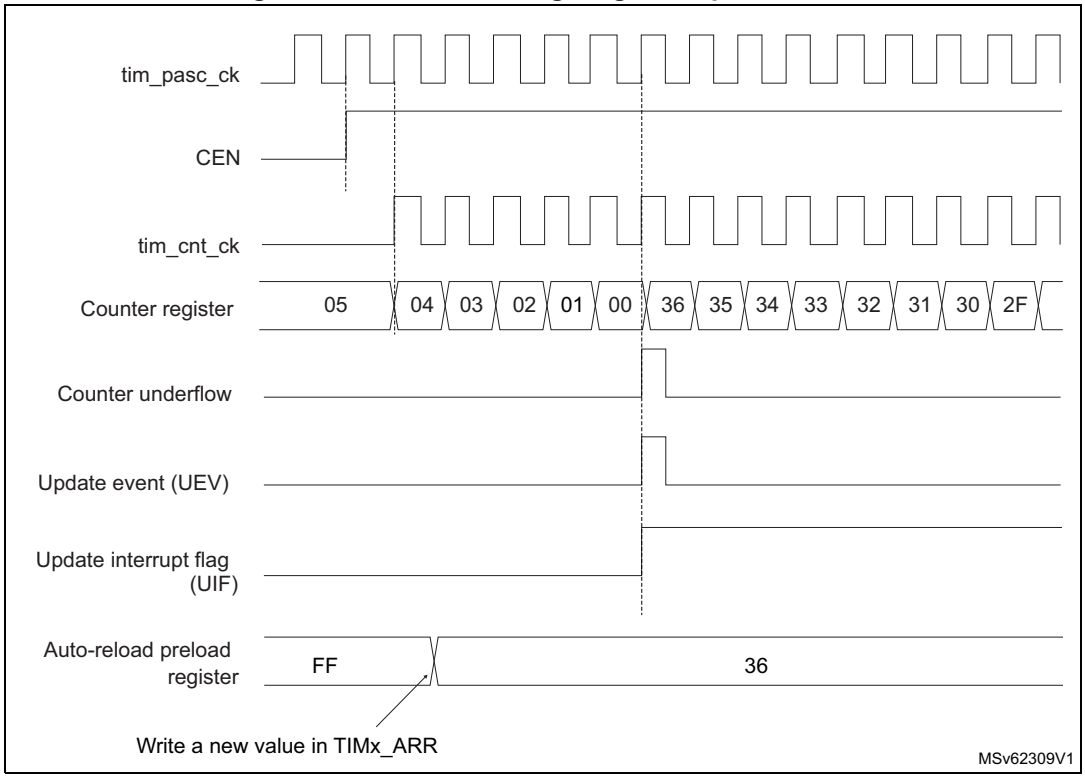


Figure 400. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

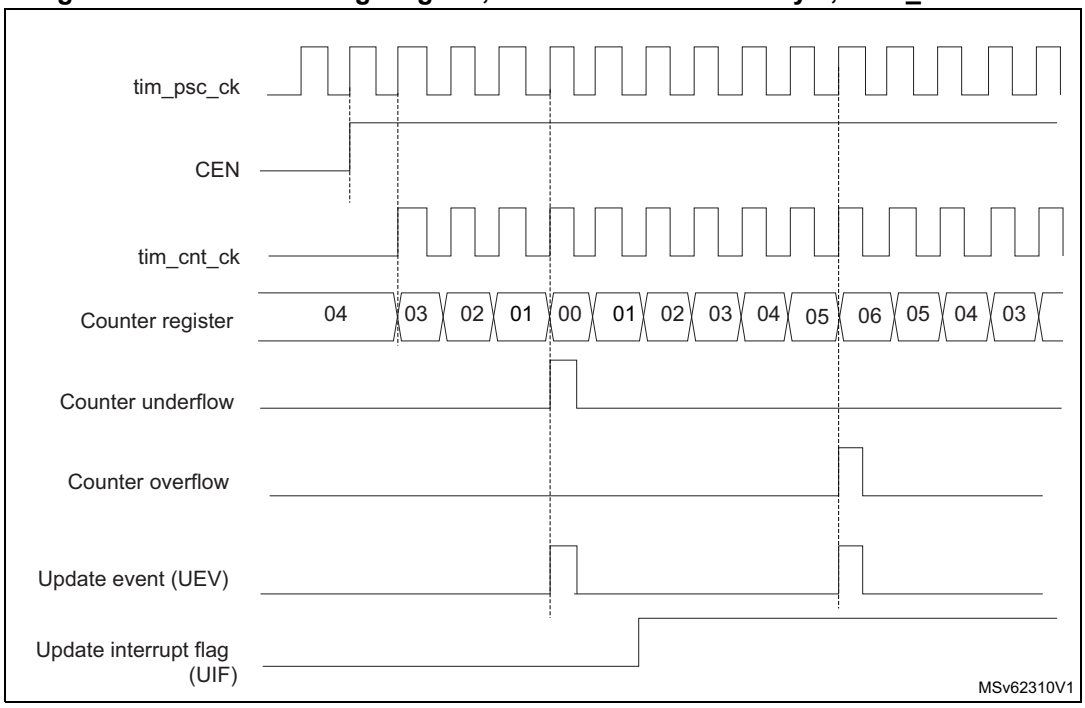
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 401. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 36.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 4)*).

Figure 402. Counter timing diagram, internal clock divided by 2

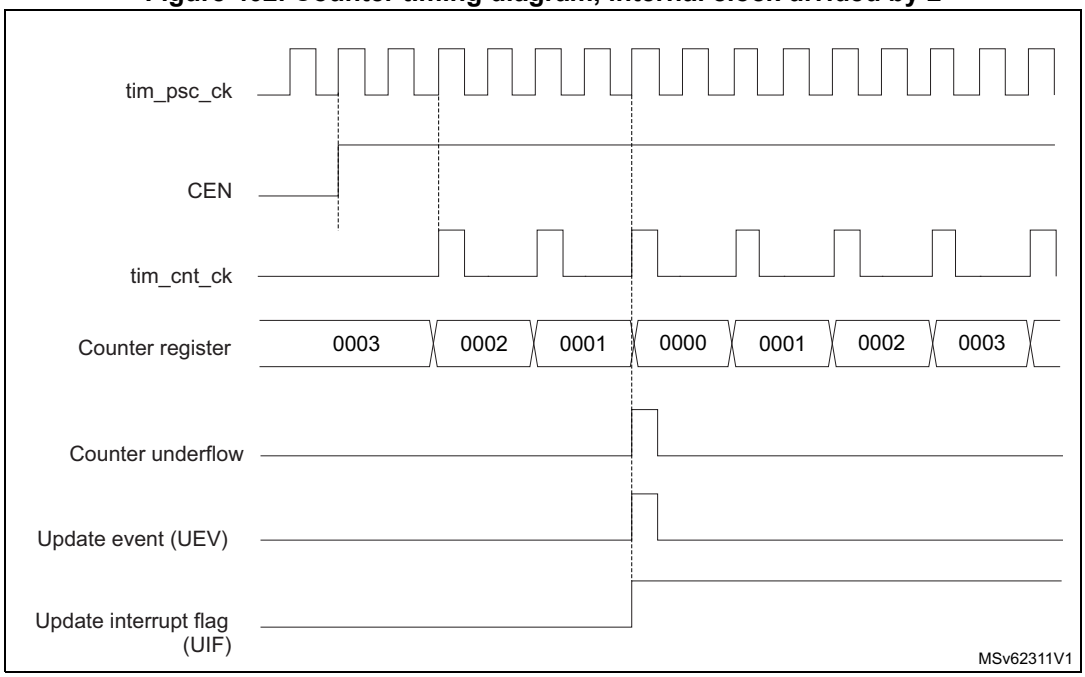
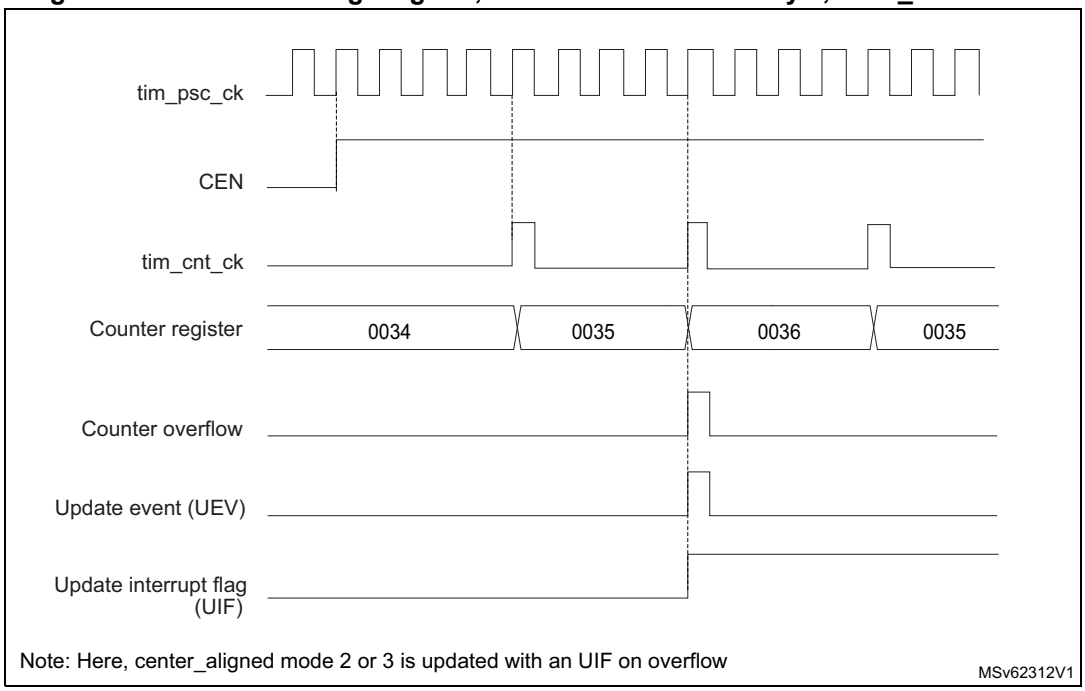


Figure 403. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 404. Counter timing diagram, internal clock divided by N

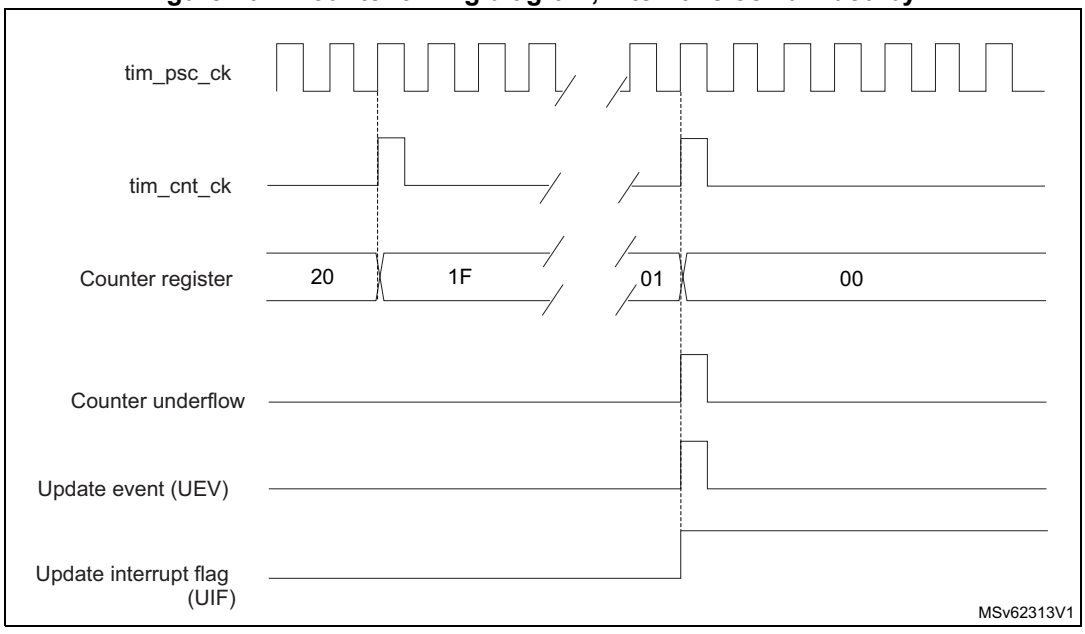


Figure 405. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

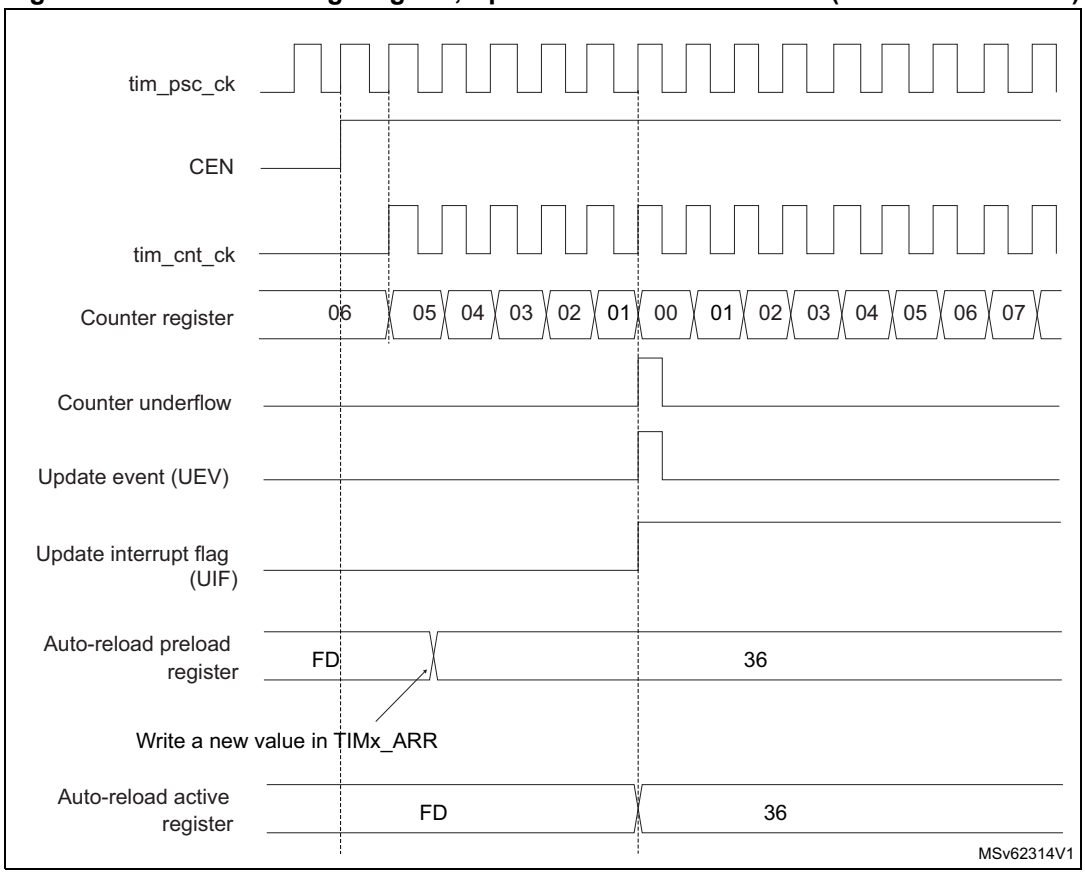
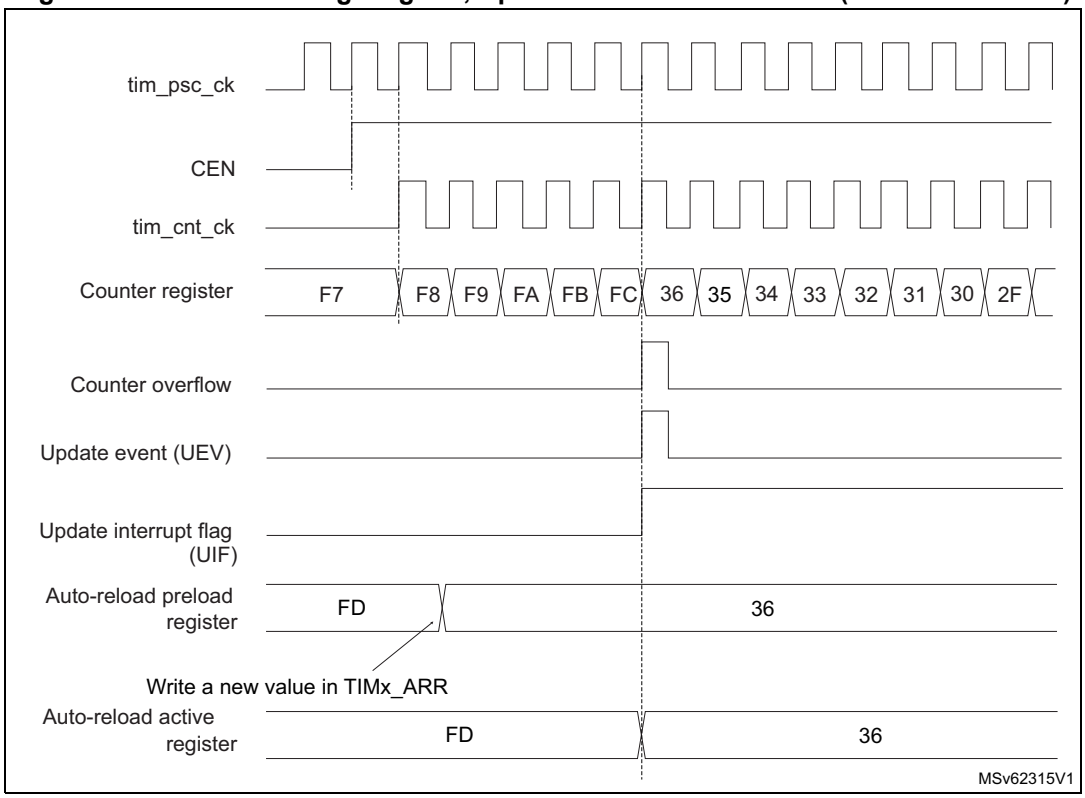


Figure 406. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



36.4.5 Clock selection

The counter clock can be provided by the following clock sources:

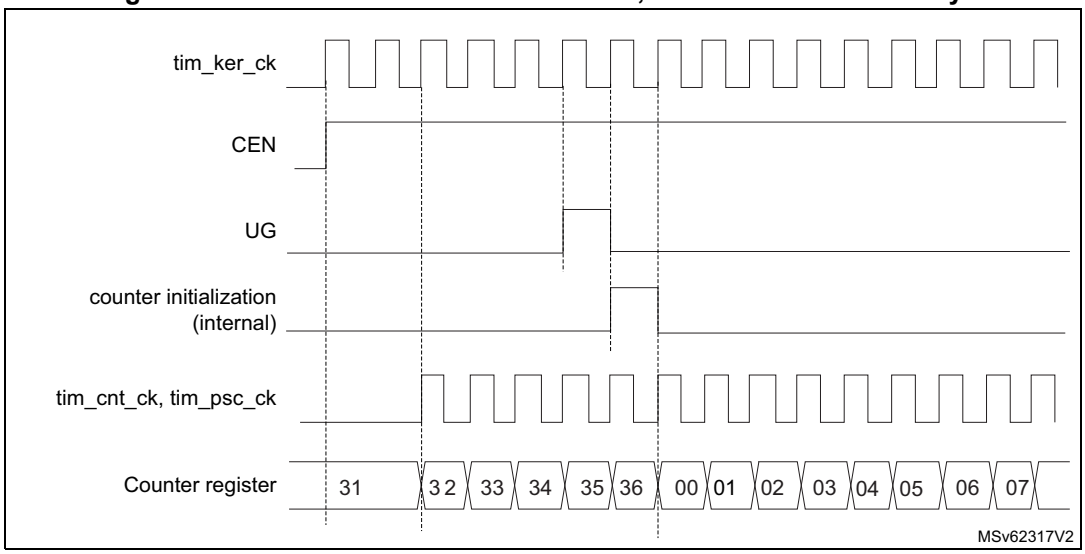
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 407 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

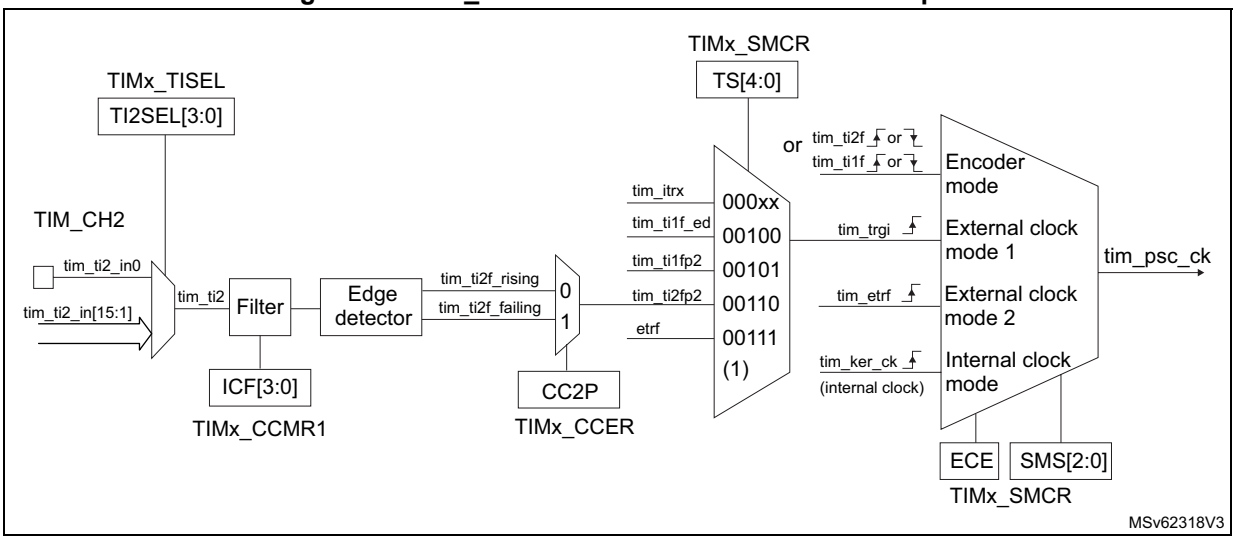
Figure 407. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 408. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).

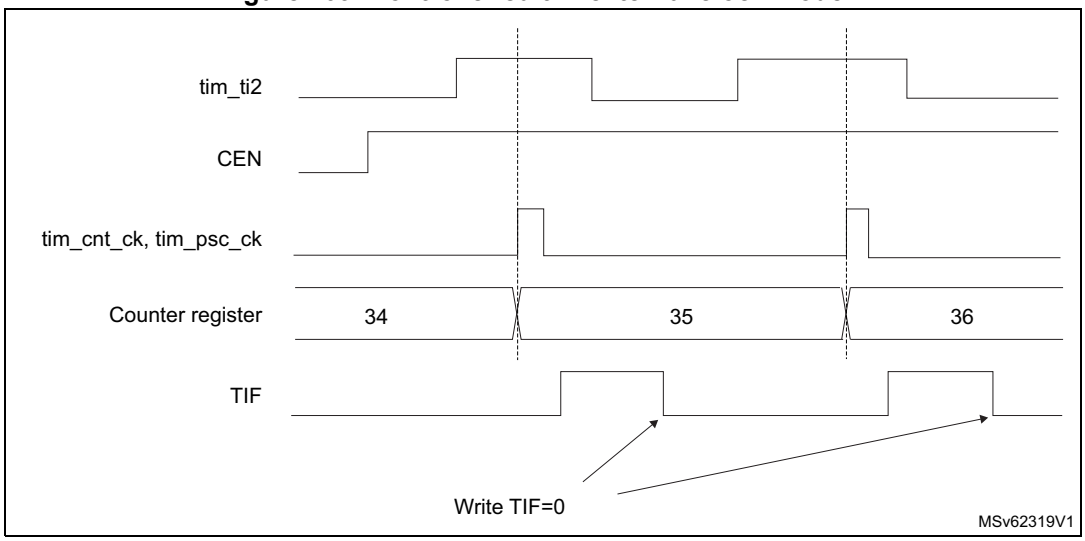
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 409. Control circuit in external clock mode 1



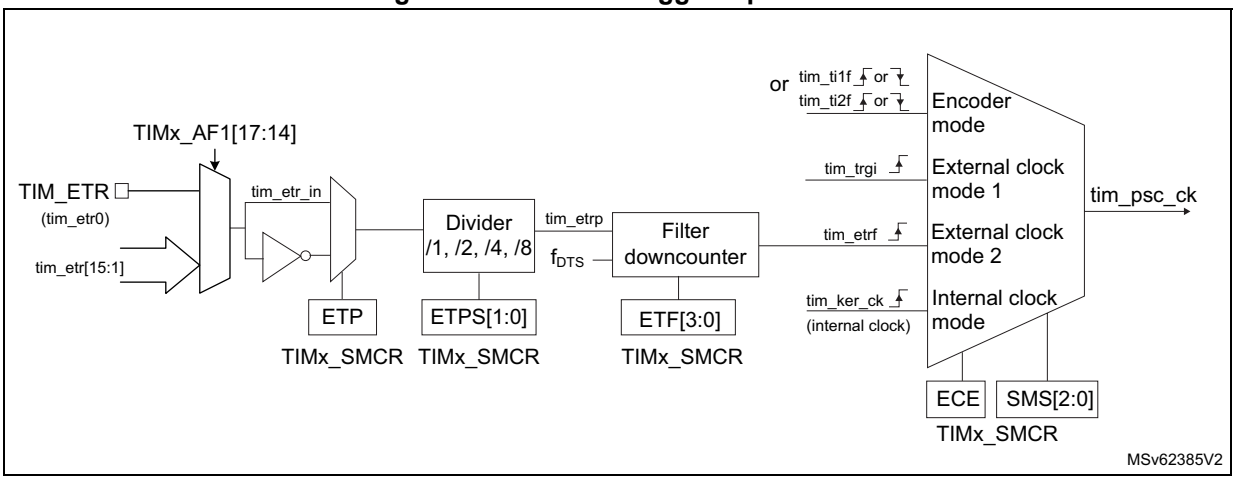
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 410 gives an overview of the external trigger input block.

Figure 410. External trigger input block



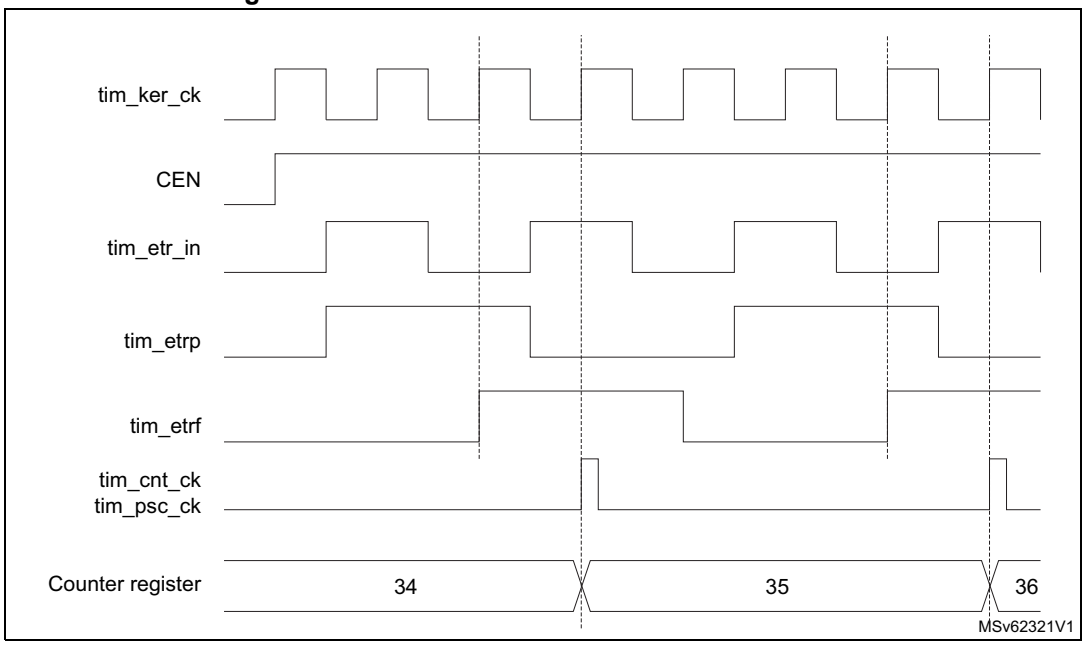
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 411. Control circuit in external clock mode 2



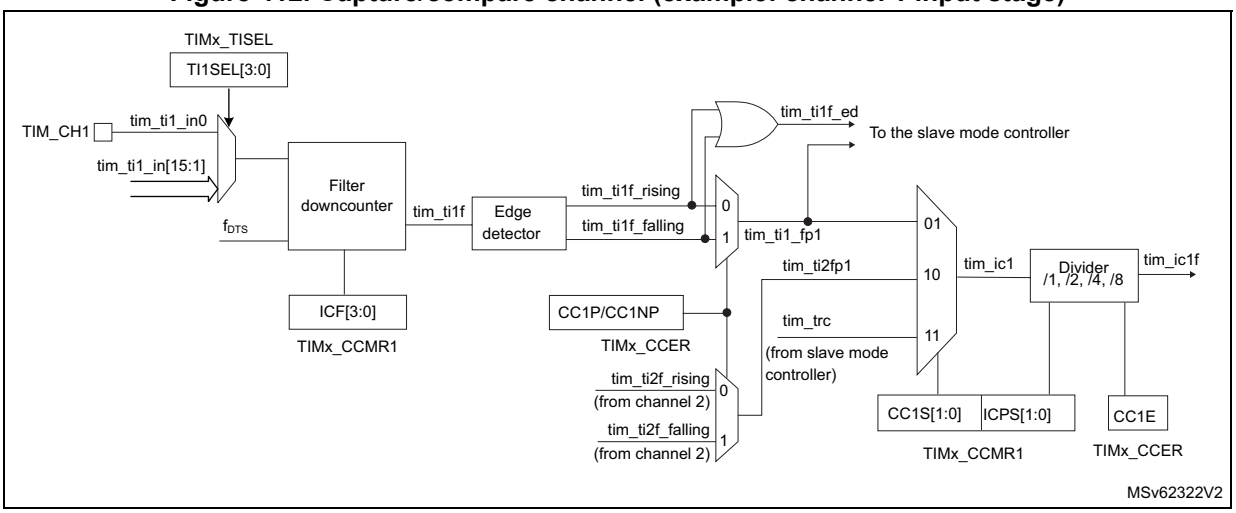
36.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 412. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 413. Capture/compare channel 1 main circuit

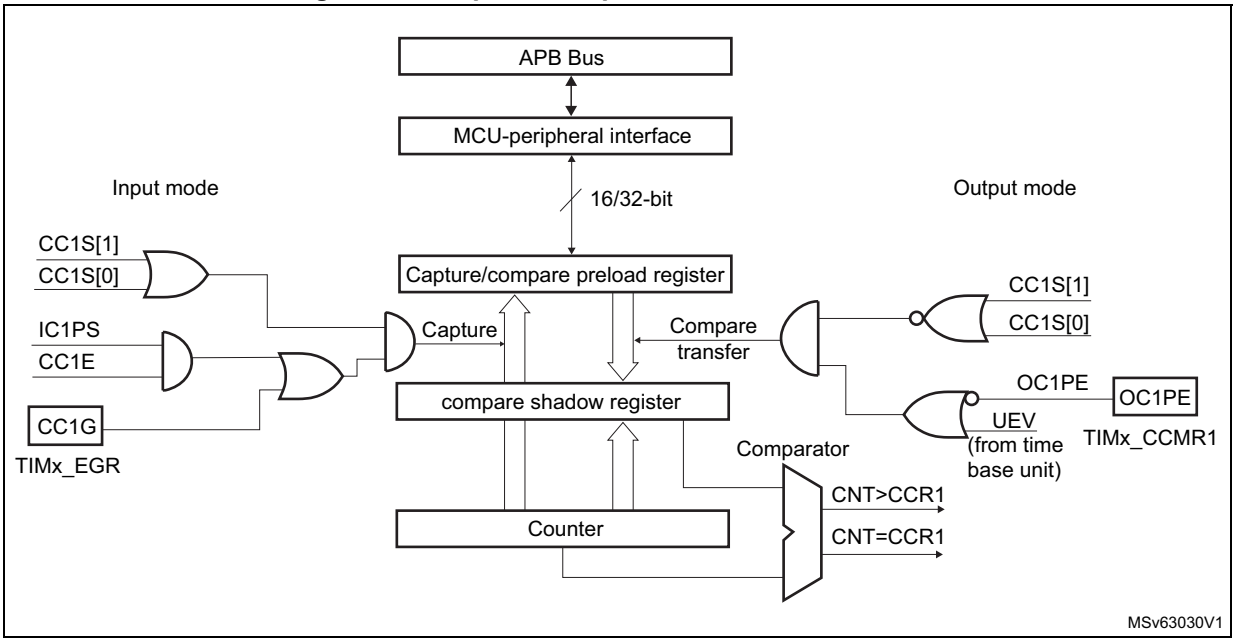
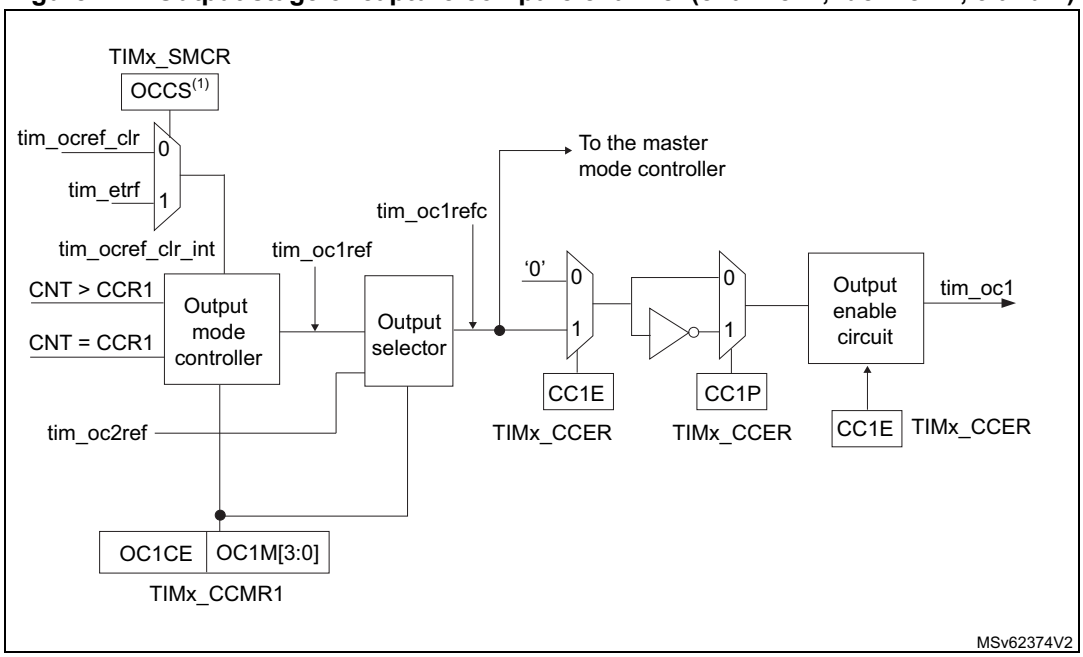


Figure 414. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

36.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

36.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

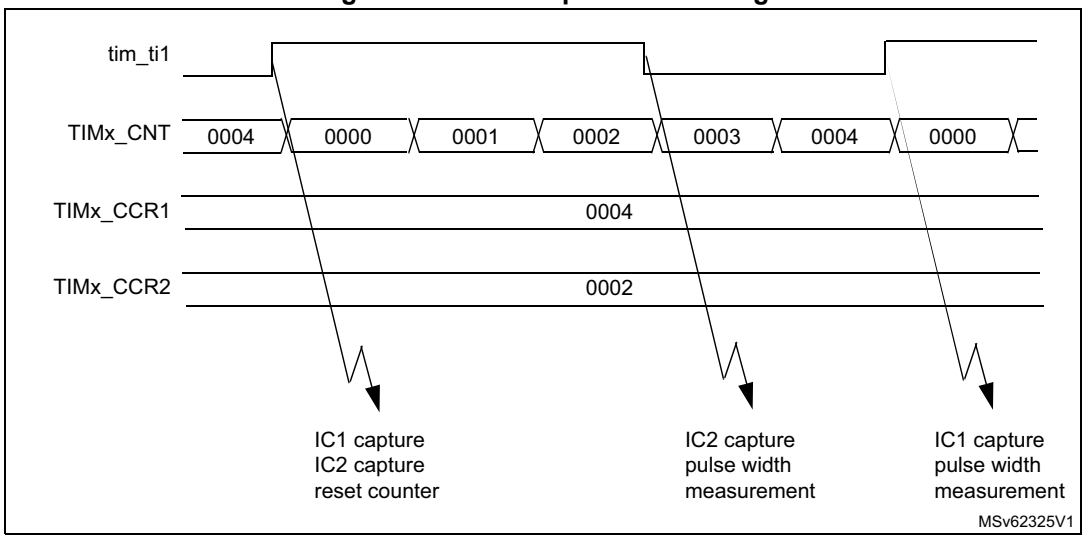
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 415. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

36.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

36.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

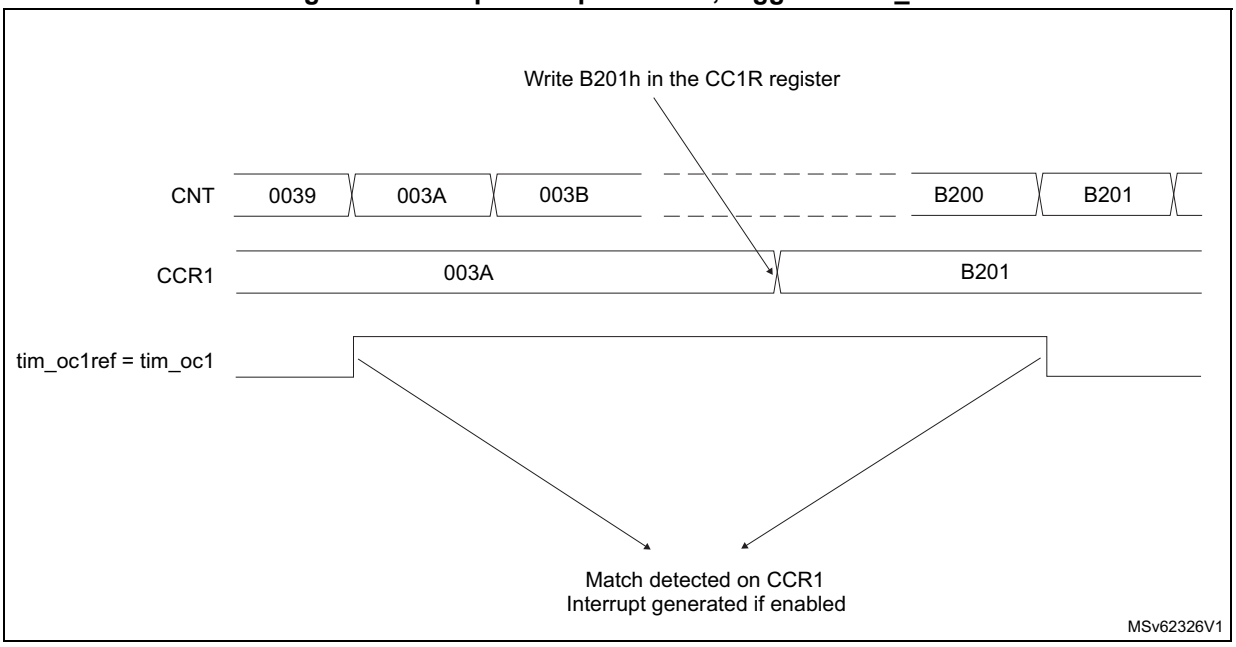
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 416*.

Figure 416. Output compare mode, toggle on tim_oc1



36.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

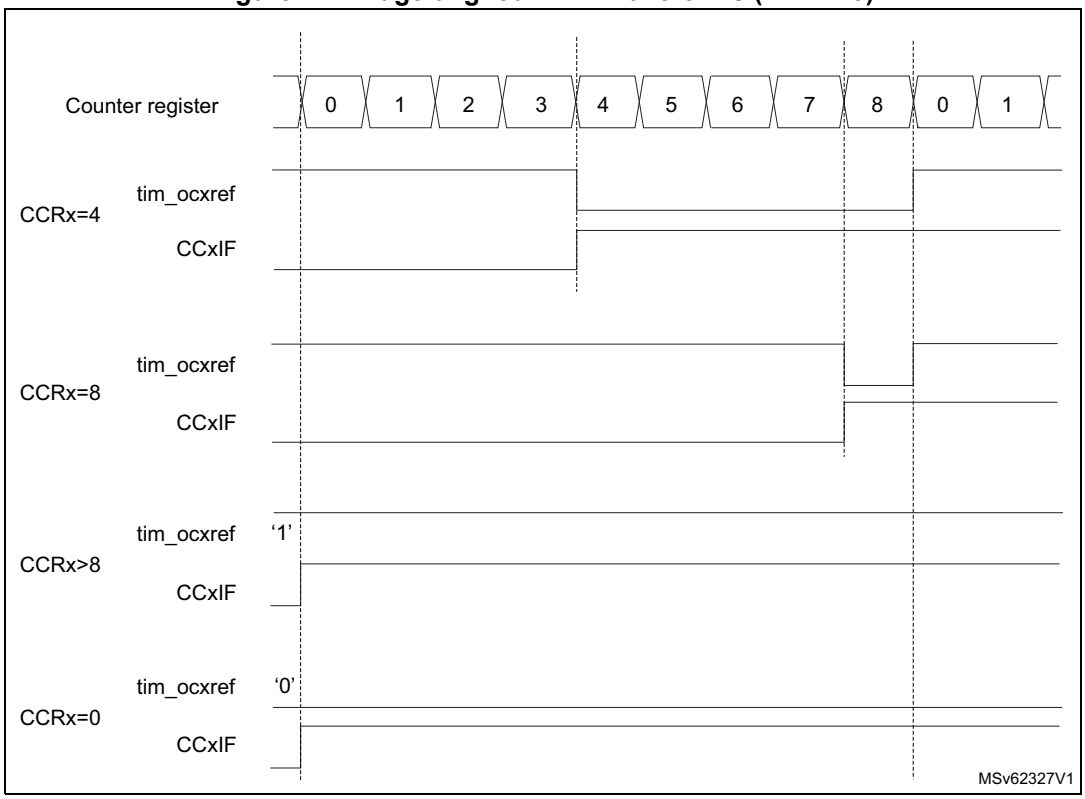
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 417* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 417. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

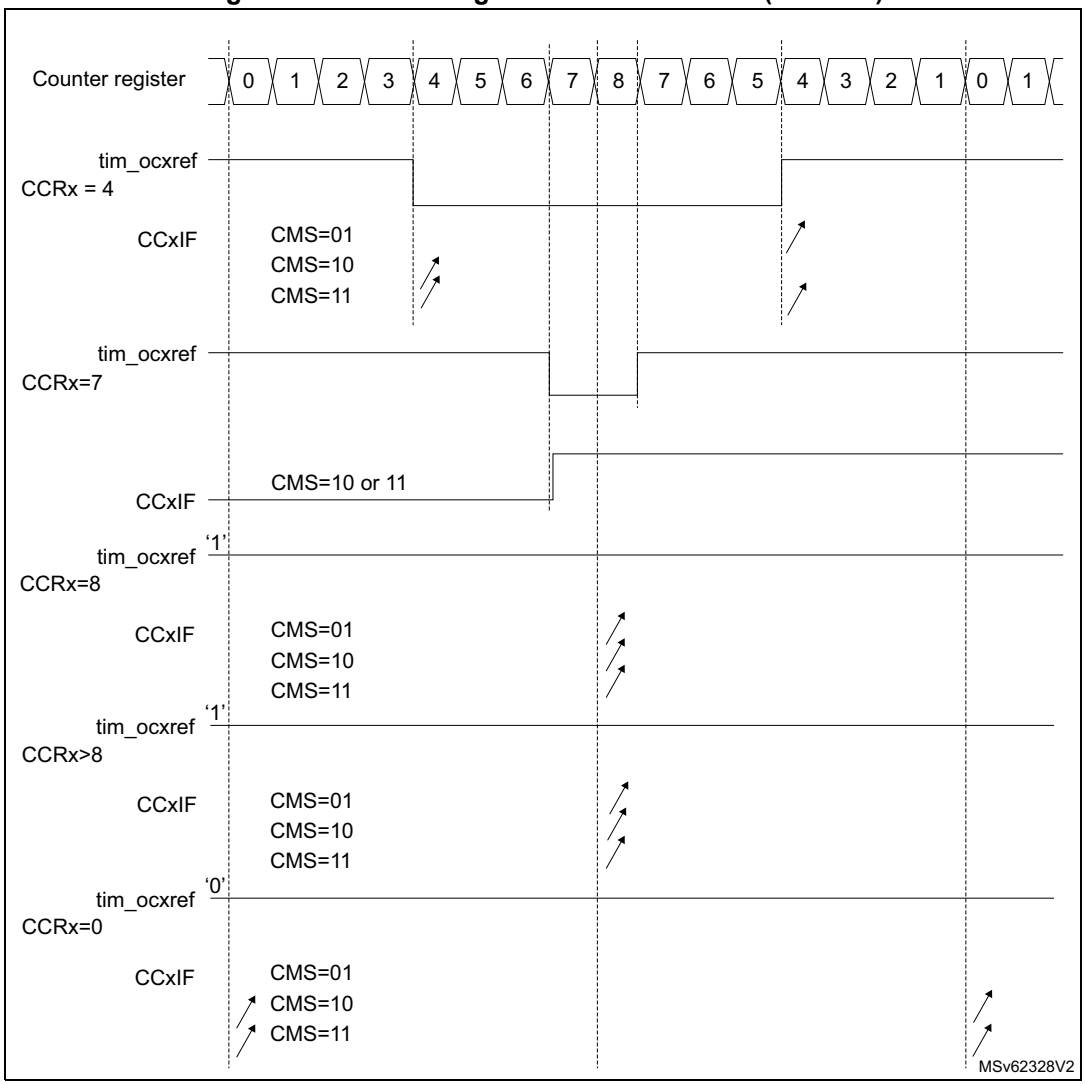
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 418 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 418. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

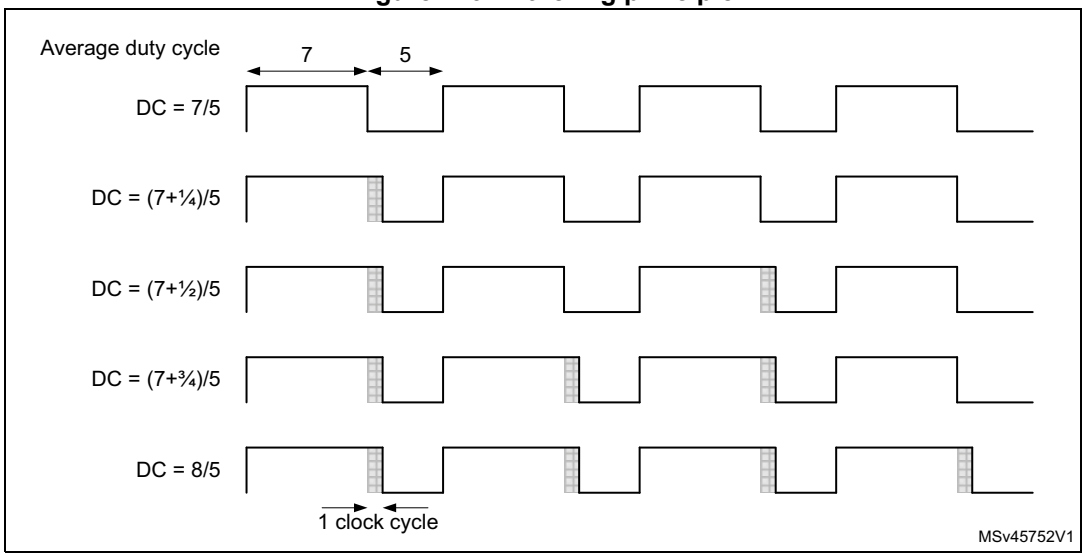
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 419* presents the dithering principle applied to four consecutive PWM cycles.

Figure 419. Dithering principle



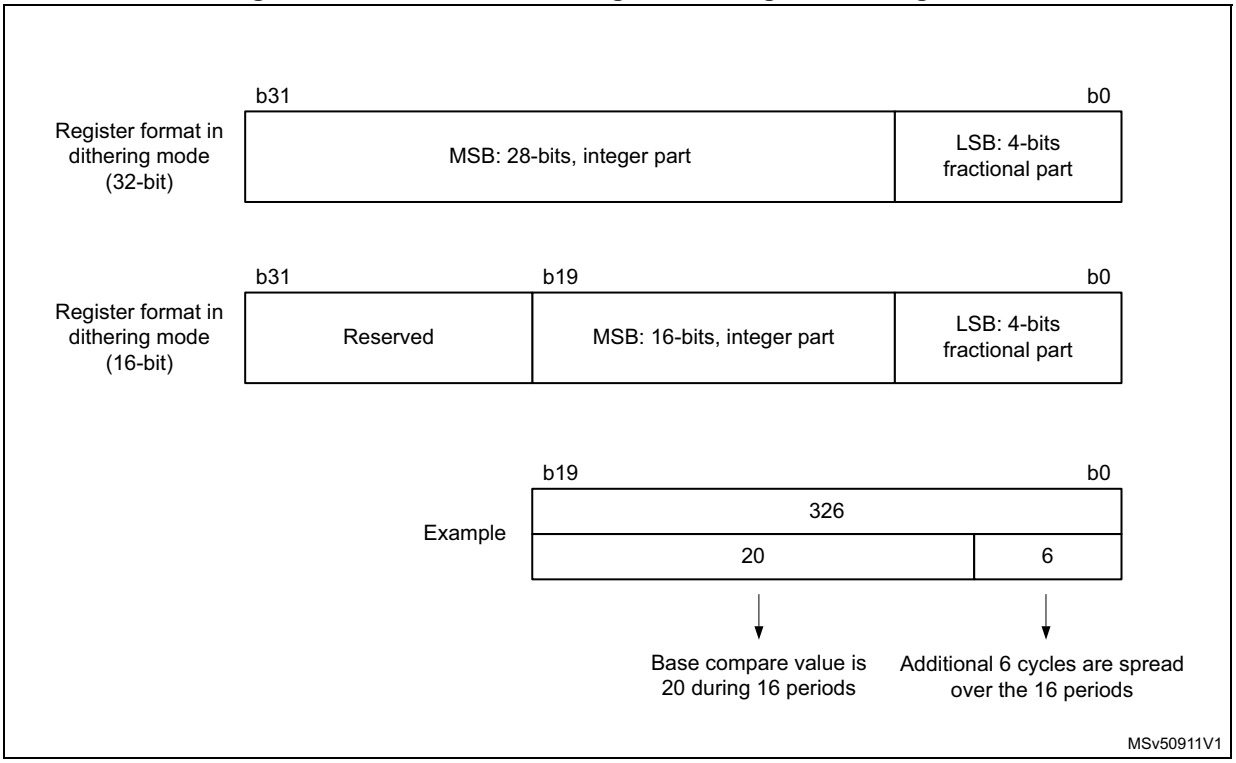
When the dithering mode is enabled, the register coding is changed as following (see *Figure 420* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 420. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 421* and *Figure 422*, the dithering mode is used to increase the PWM resolution.

Figure 421. PWM resolution vs frequency (16-bit mode)

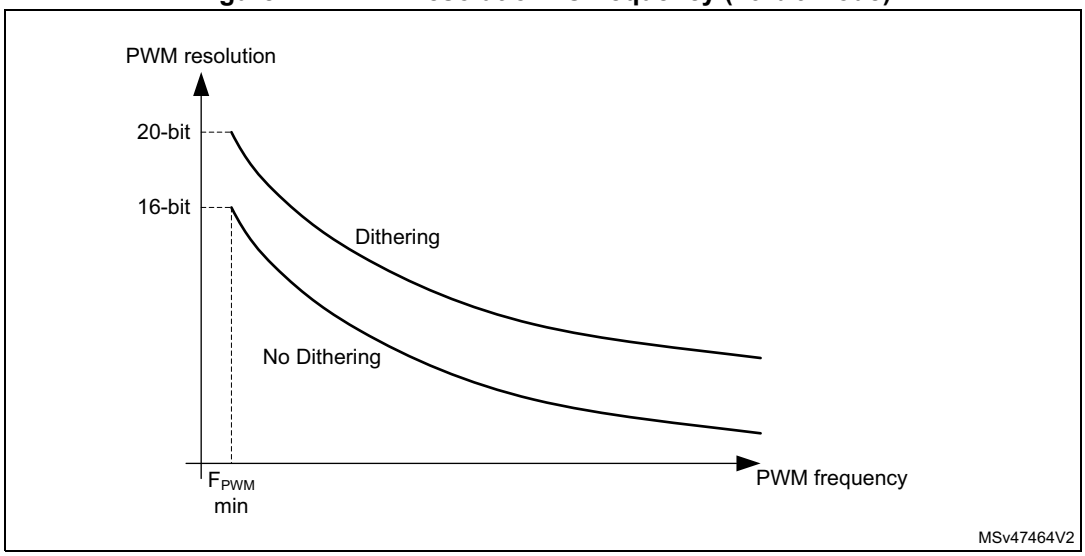
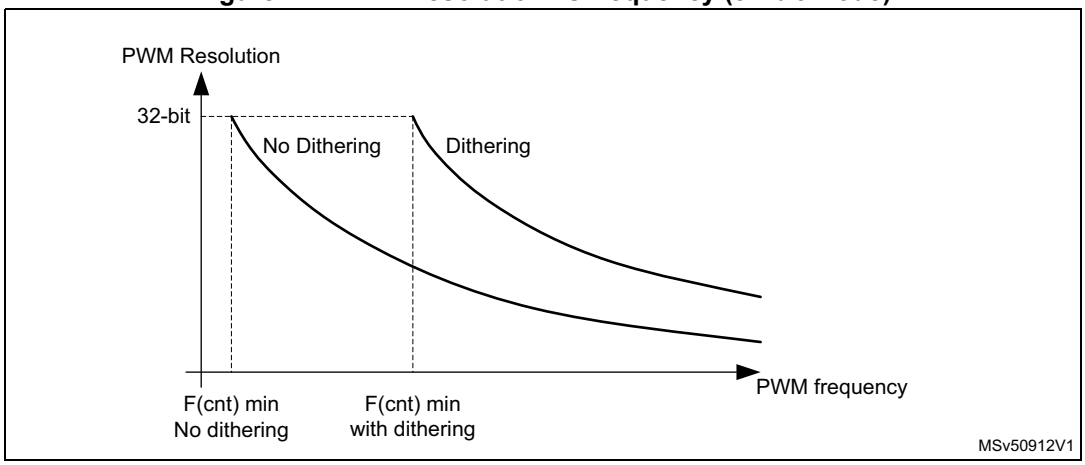
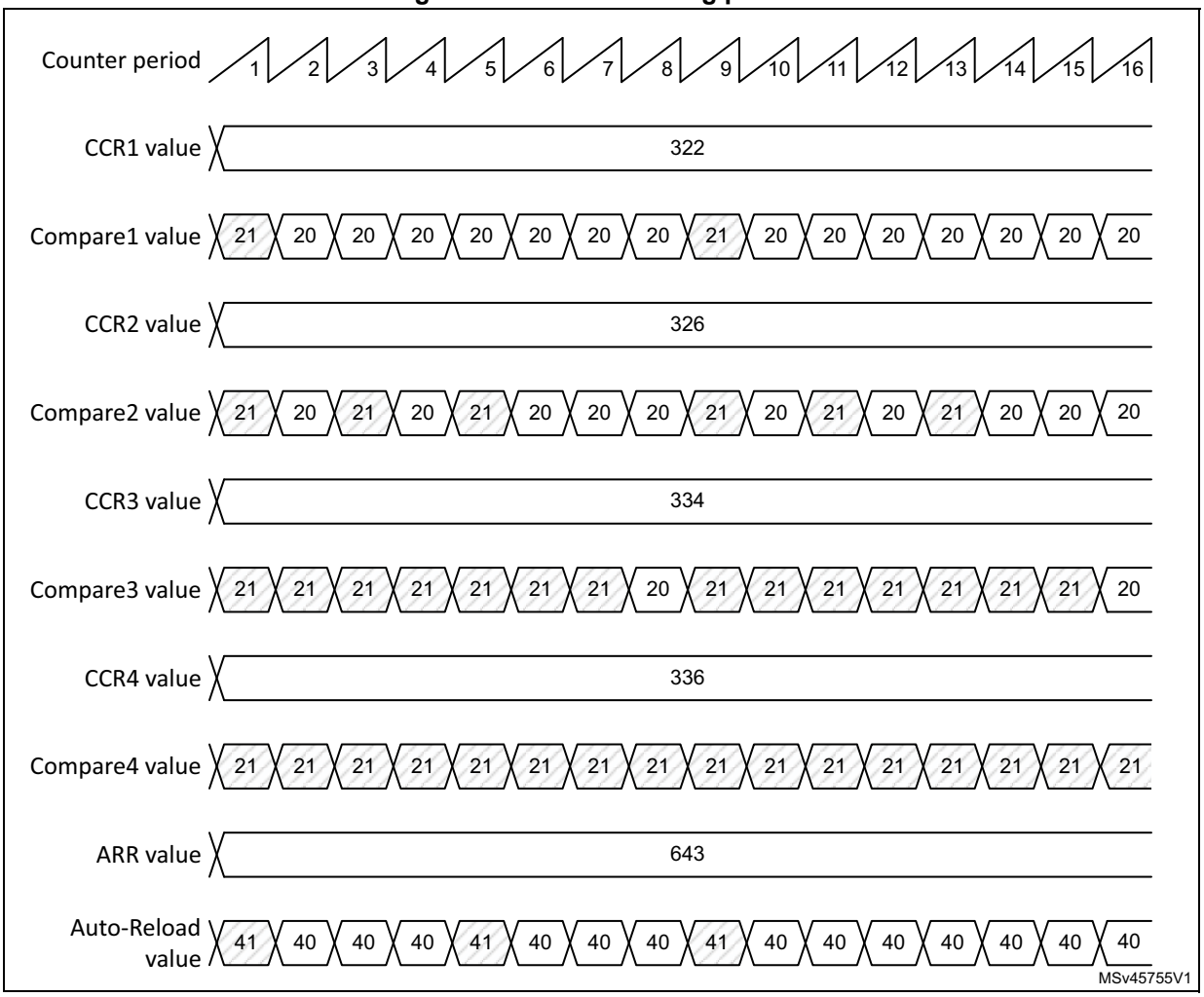


Figure 422. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 423*.

Figure 423. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 374*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 374. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 424*.

Figure 424. Dithering effect on duty cycle in center-aligned PWM mode

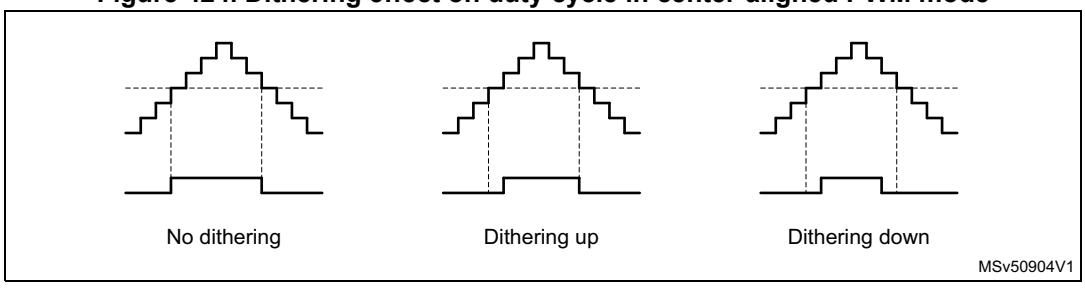


Table 375 shows how the dithering pattern is added in center-aligned PWM mode.

Table 375. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

36.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

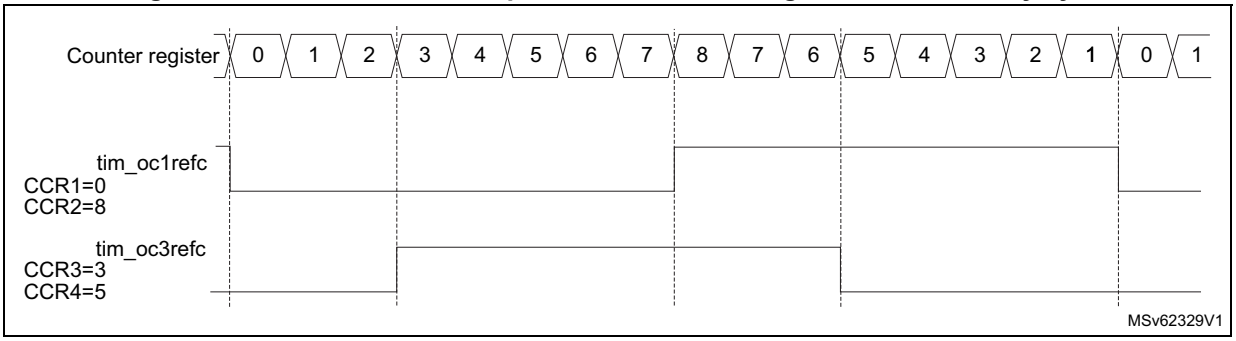
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 425 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 425. Generation of two phase-shifted PWM signals with 50% duty cycle



36.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

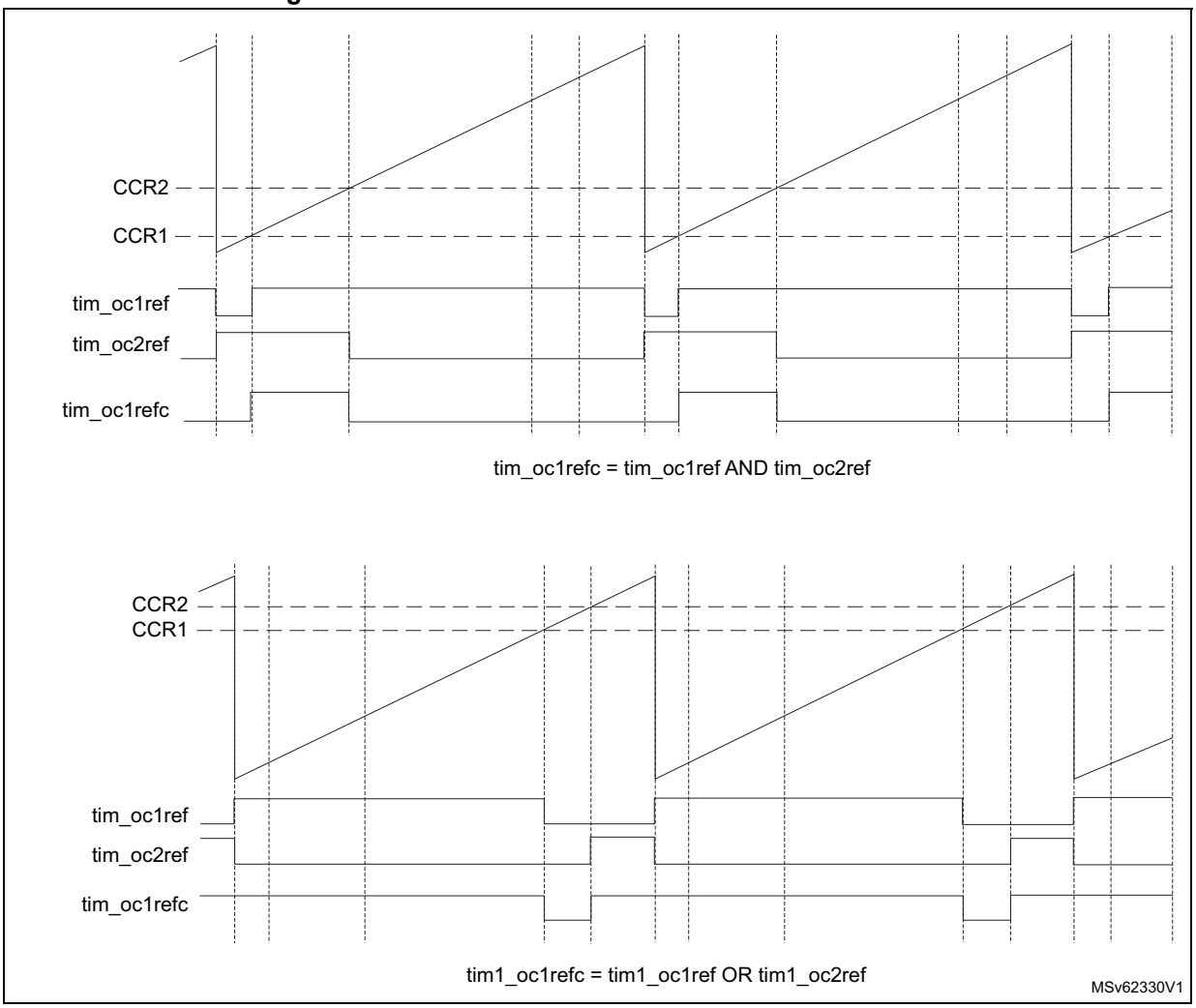
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 426 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 426. Combined PWM mode on channels 1 and 3



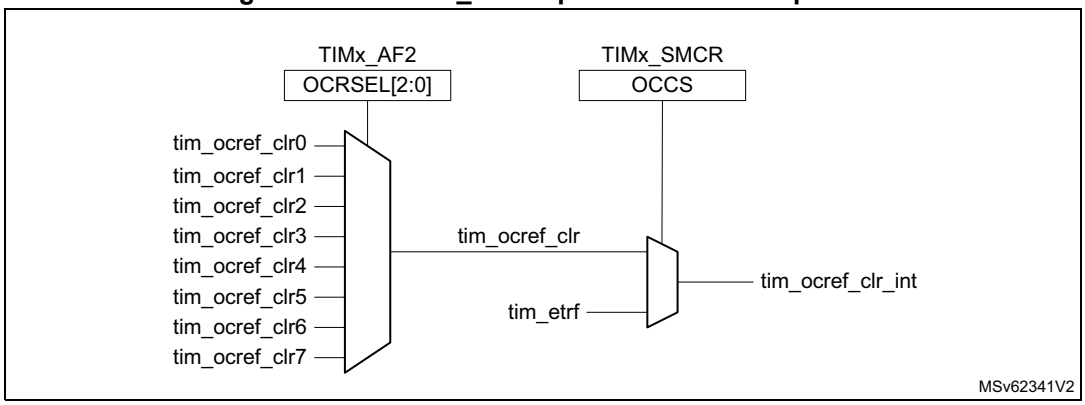
36.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 36.3: TIM2/TIM3/TIM4 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 427*.

Figure 427. OCREF_CLR input selection multiplexer



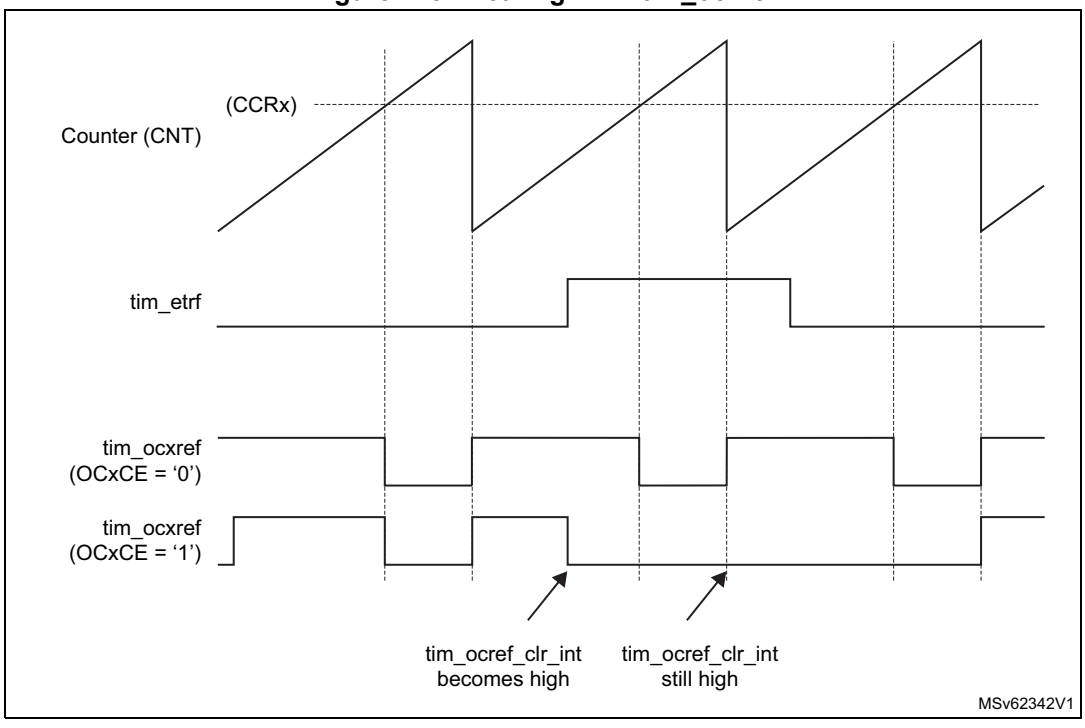
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 428 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 428. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

36.4.15 One-pulse mode

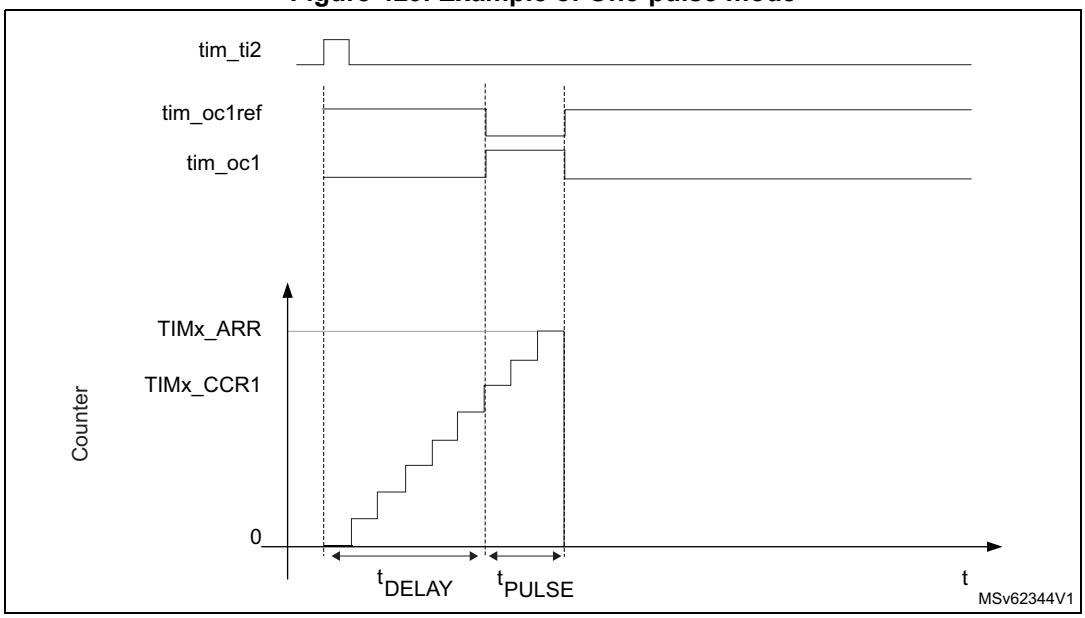
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 429. Example of One-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

36.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 36.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

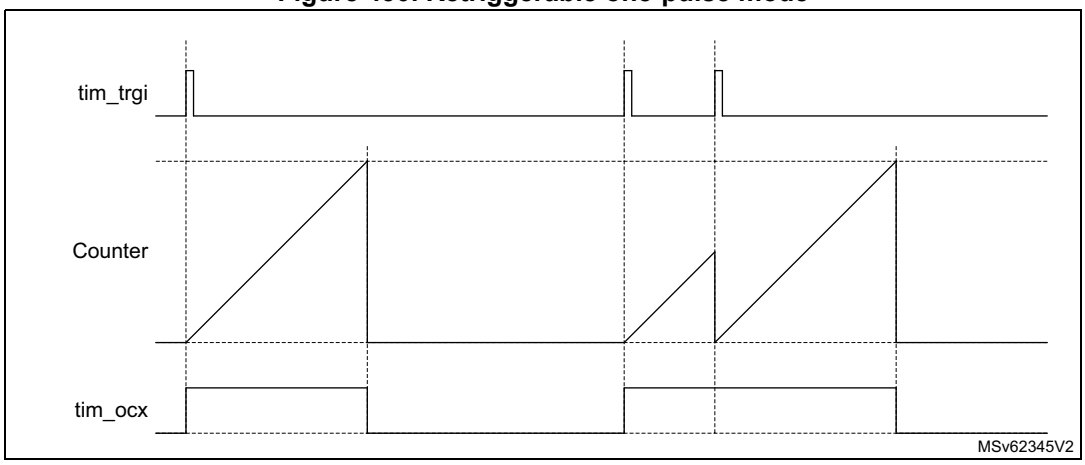
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 430. Retriggerable one-pulse mode



36.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 431*.

Figure 431. Pulse generator circuitry

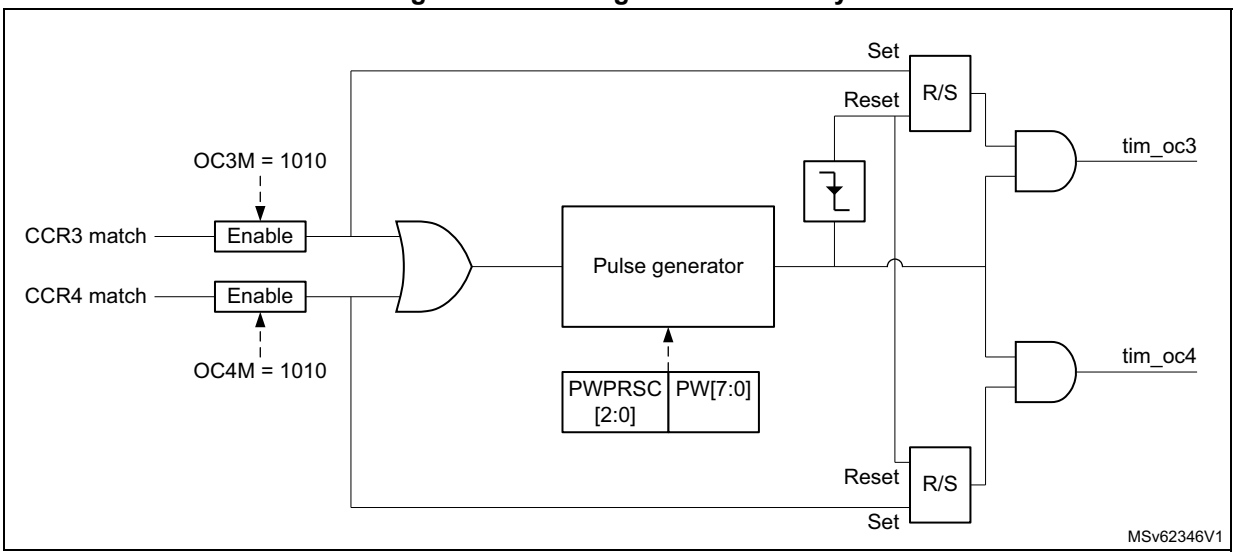
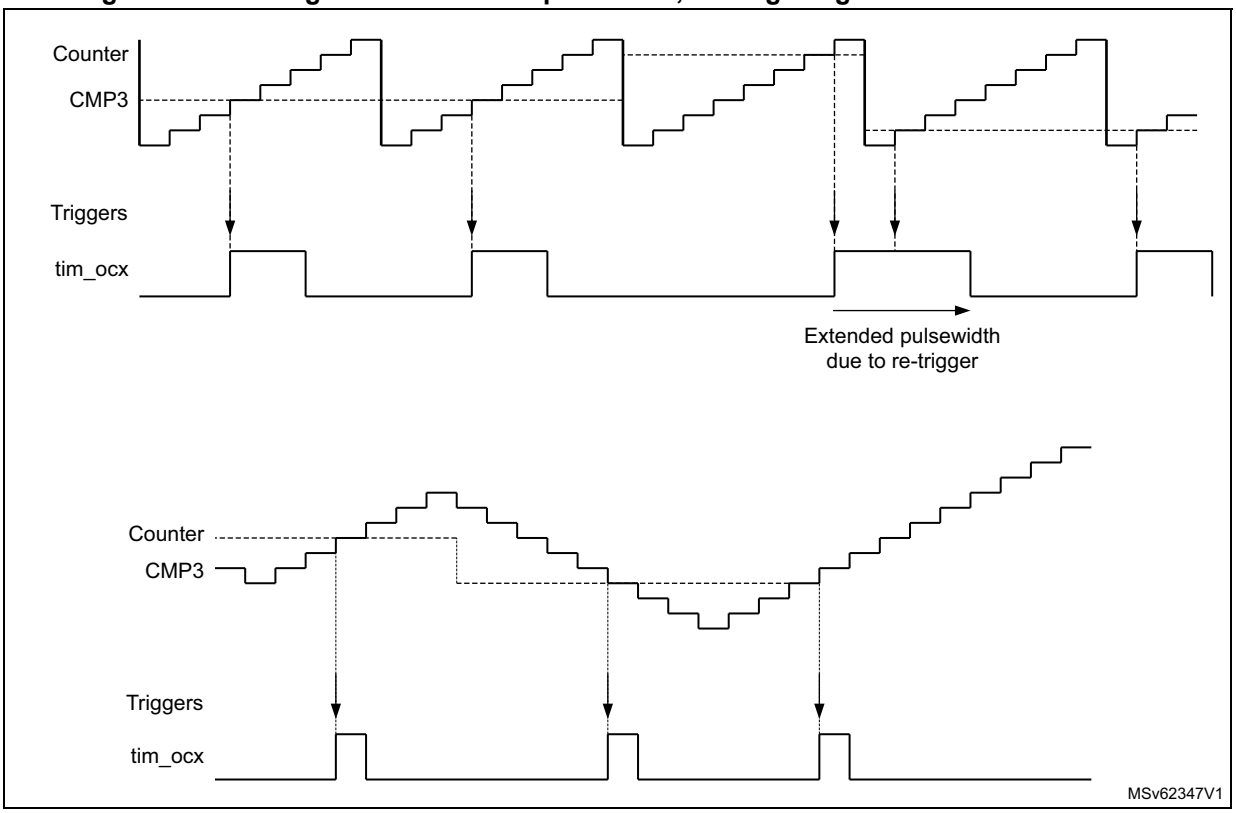


Figure 432 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 432. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

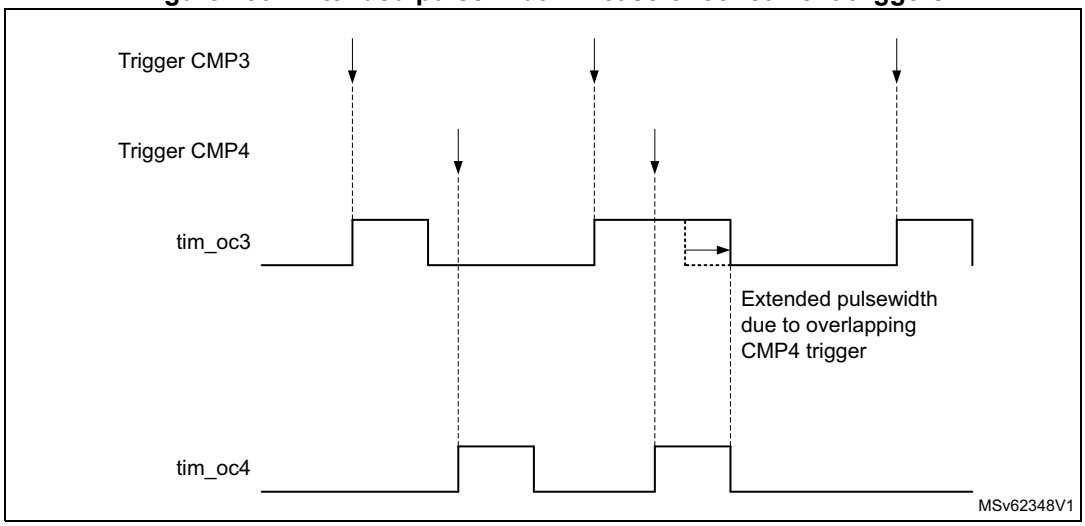
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 433).

Figure 433. Extended pulse width in case of concurrent triggers



36.4.18 Encoder interface mode

Quadrature encoder

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 376*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or

tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 376. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 434 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 434. Example of counter operation in encoder interface mode

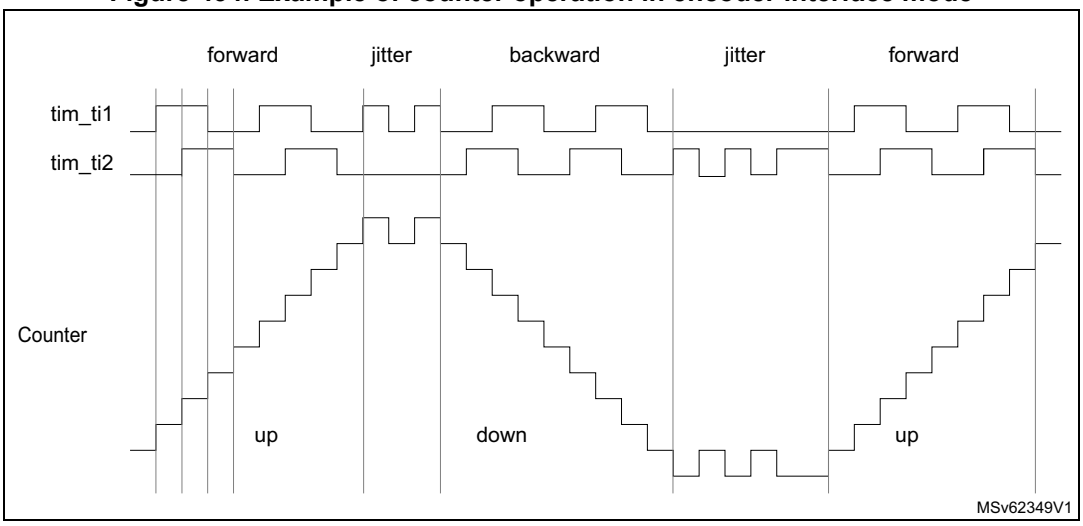


Figure 435 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 435. Example of encoder interface mode with tim_ti1fp1 polarity inverted

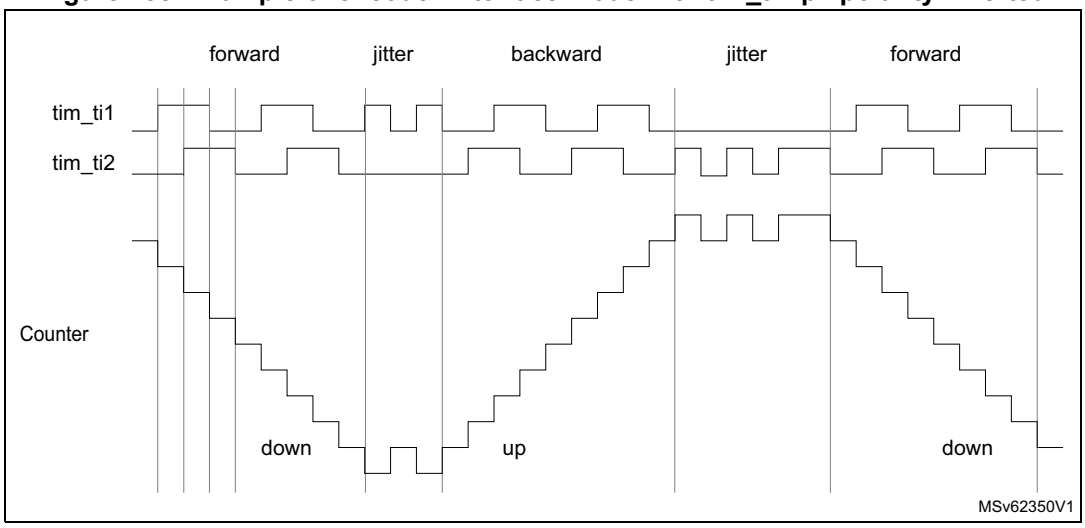
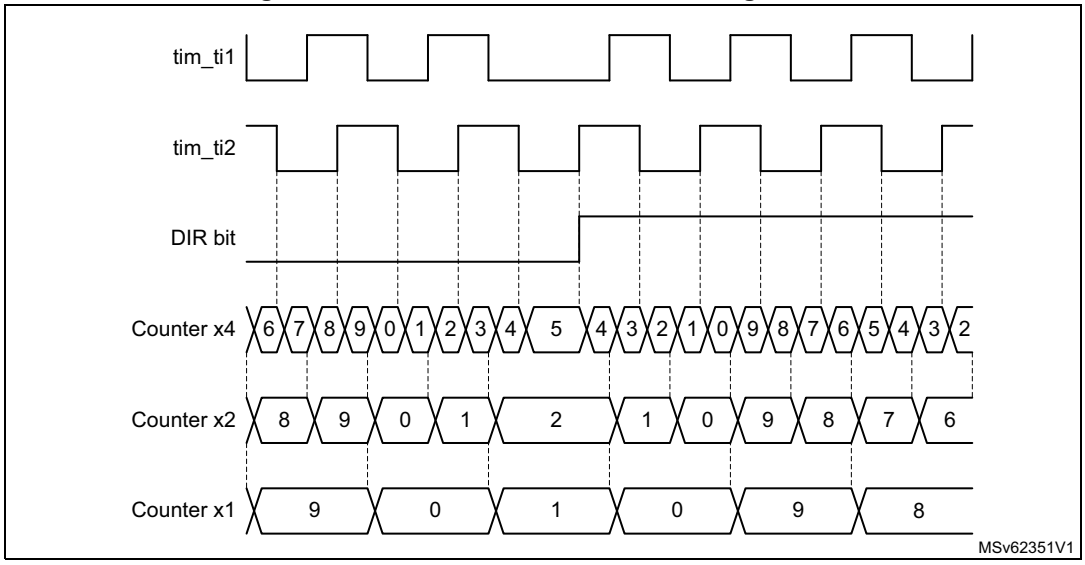


Figure 436 shows the timer counter value during a speed reversal, for various counting modes.

Figure 436. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

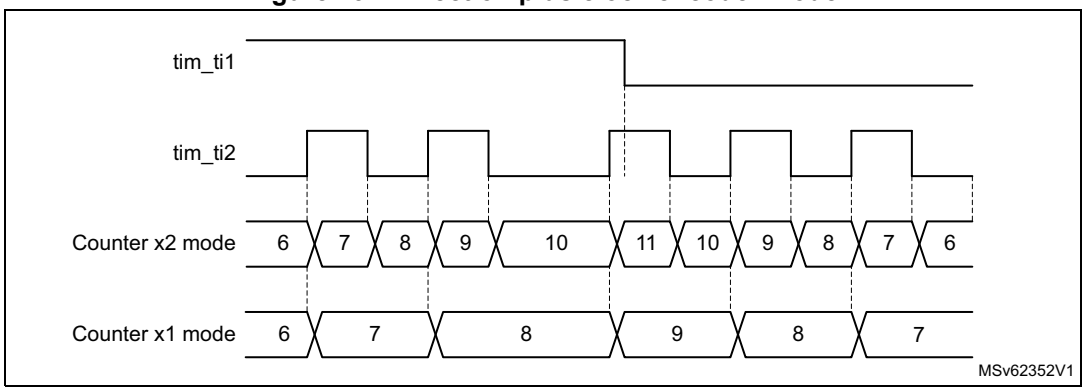
In the "clock plus direction" mode shown on *Figure 437*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 437. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 438*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 438*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 439*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 438*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 439*).

Figure 438. Directional clock encoder mode (CC1P = CC2P = 0)

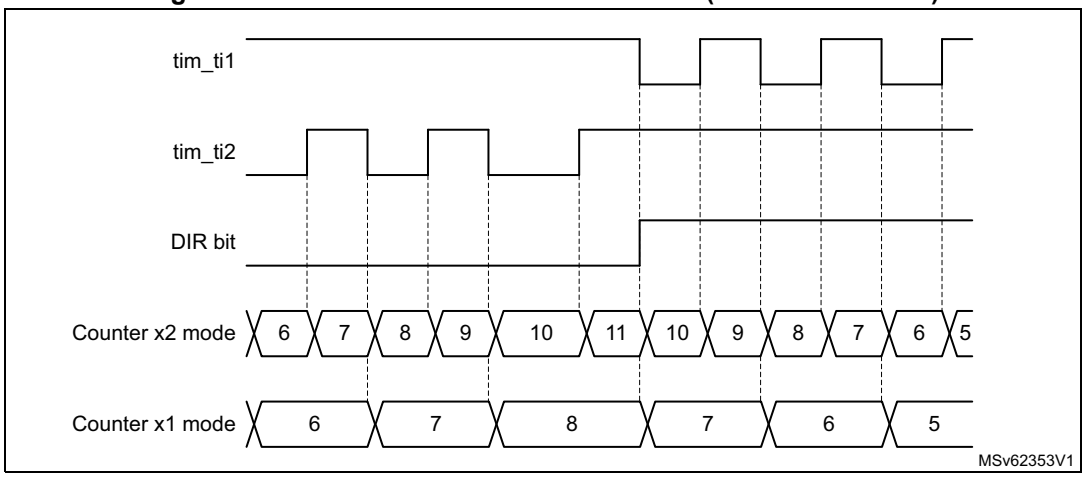


Figure 439. Directional clock encoder mode (CC1P = CC2P = 1)

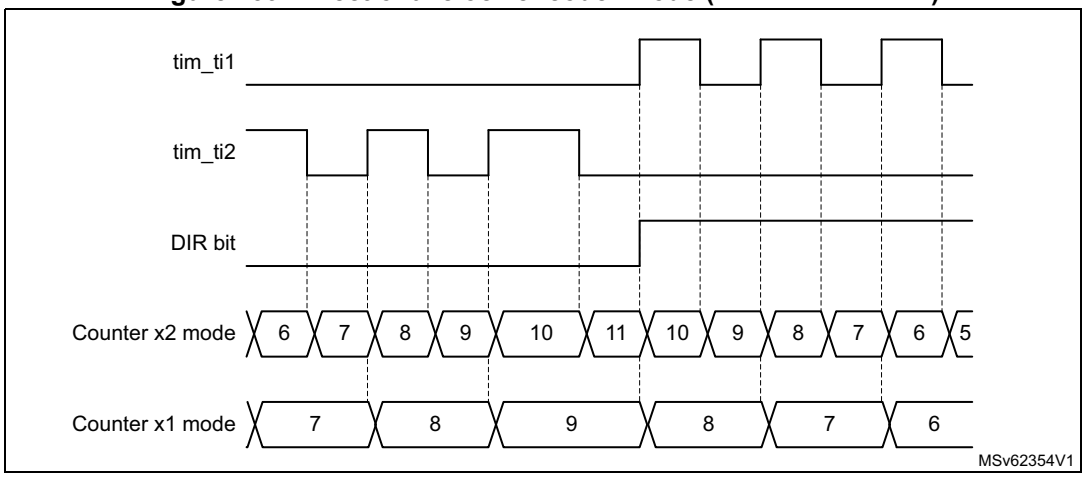


Table 377 details how the directional clock mode operates, for any input transition.

Table 377. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

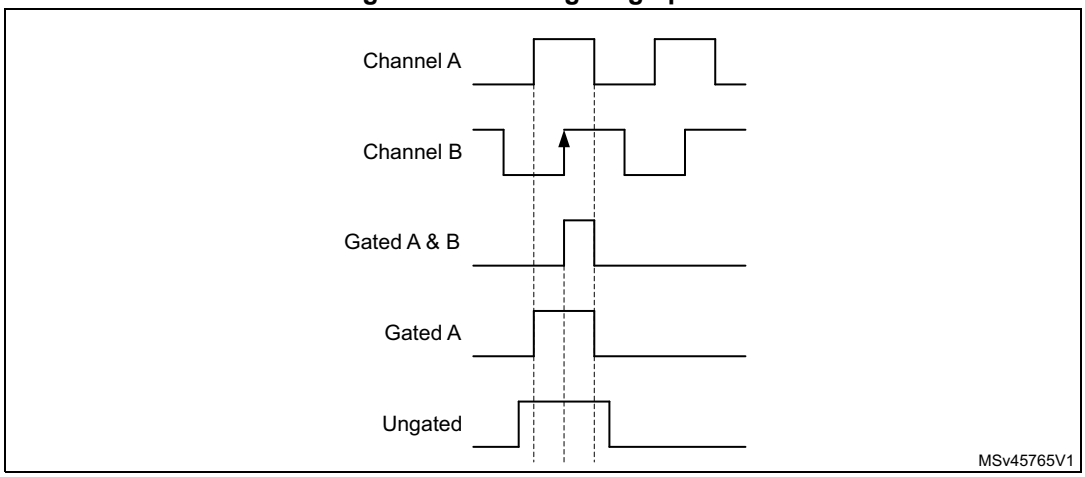
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 440*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

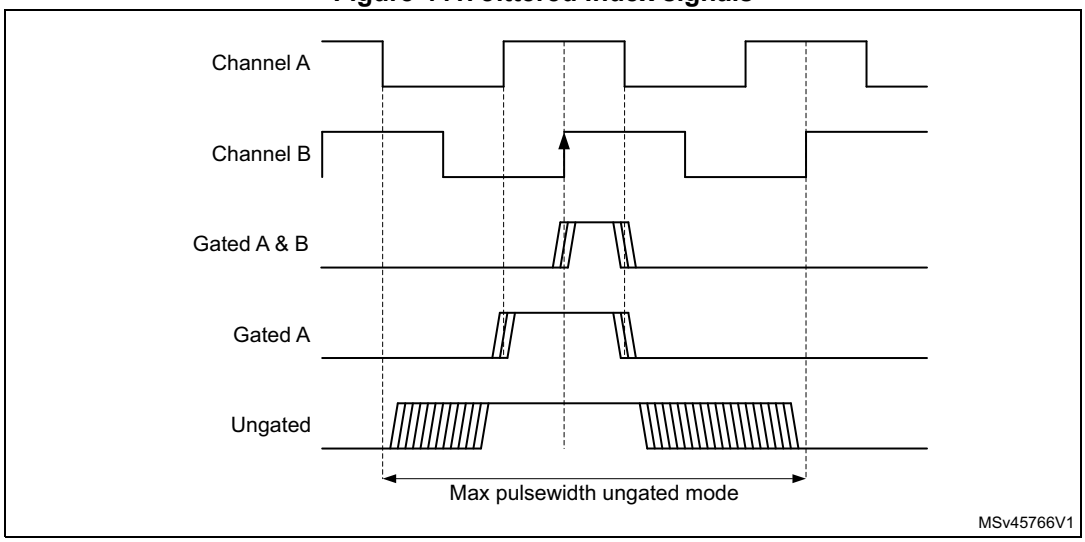
Figure 440. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 441*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 441. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 442* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 442. Index generation for IPOS[1:0] = 11

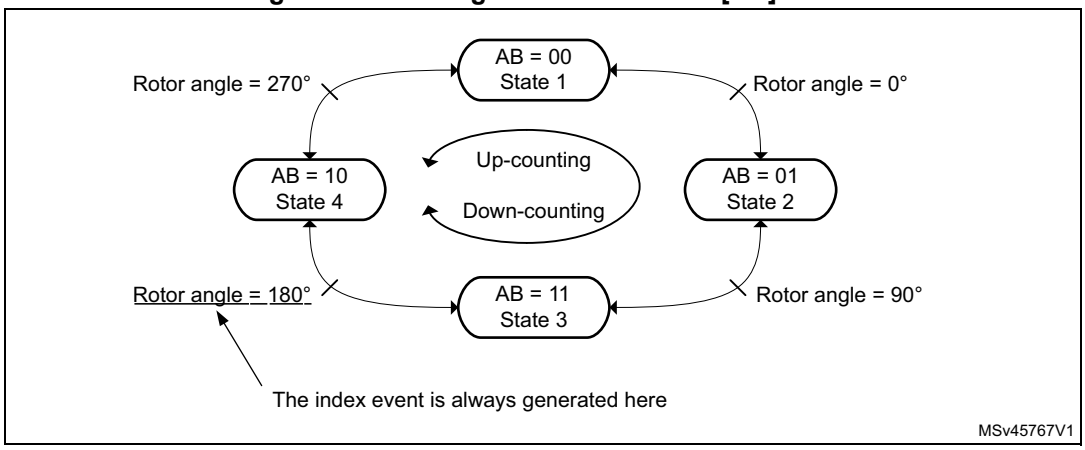


Figure 443 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 443. Counter reading with index gated on channel A (IPOS[1:0] = 11)

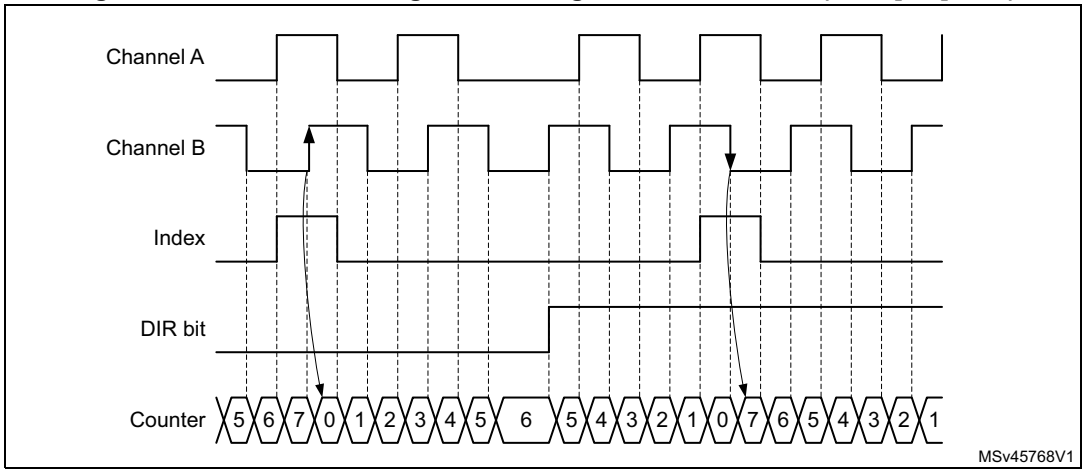


Figure 444 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 444. Counter reading with index ungated (IPOS[1:0] = 00)

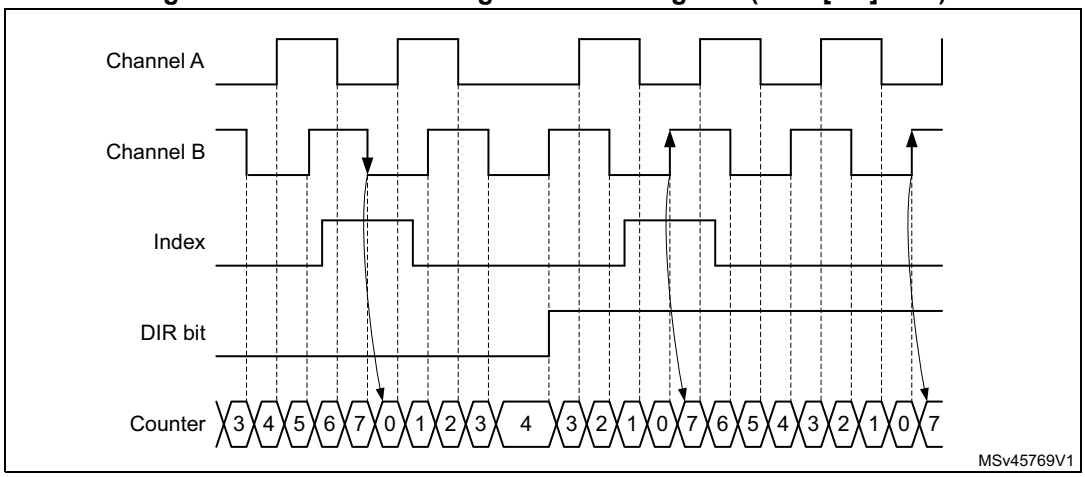


Figure 445 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 445. Counter reading with index gated on channel A and B

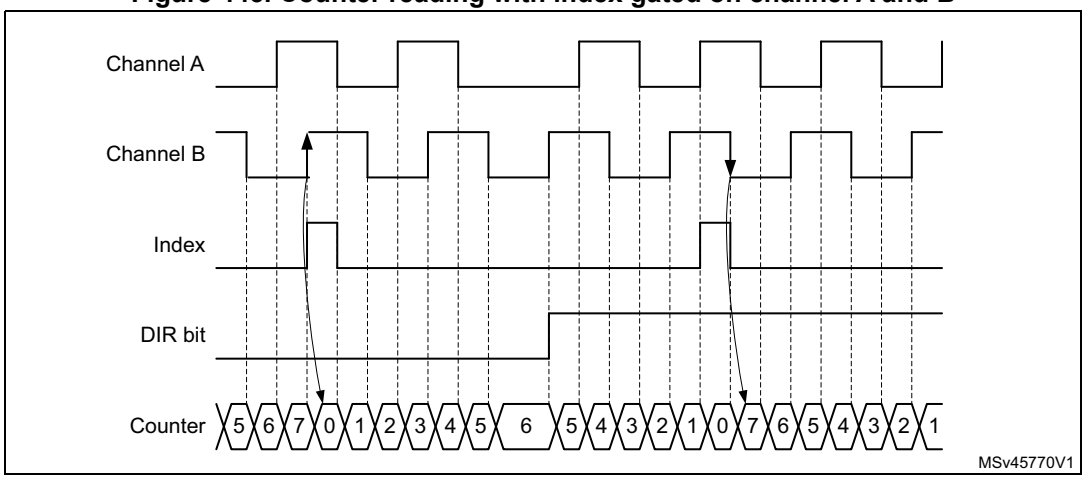


Figure 446 and *Figure 447* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 446. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

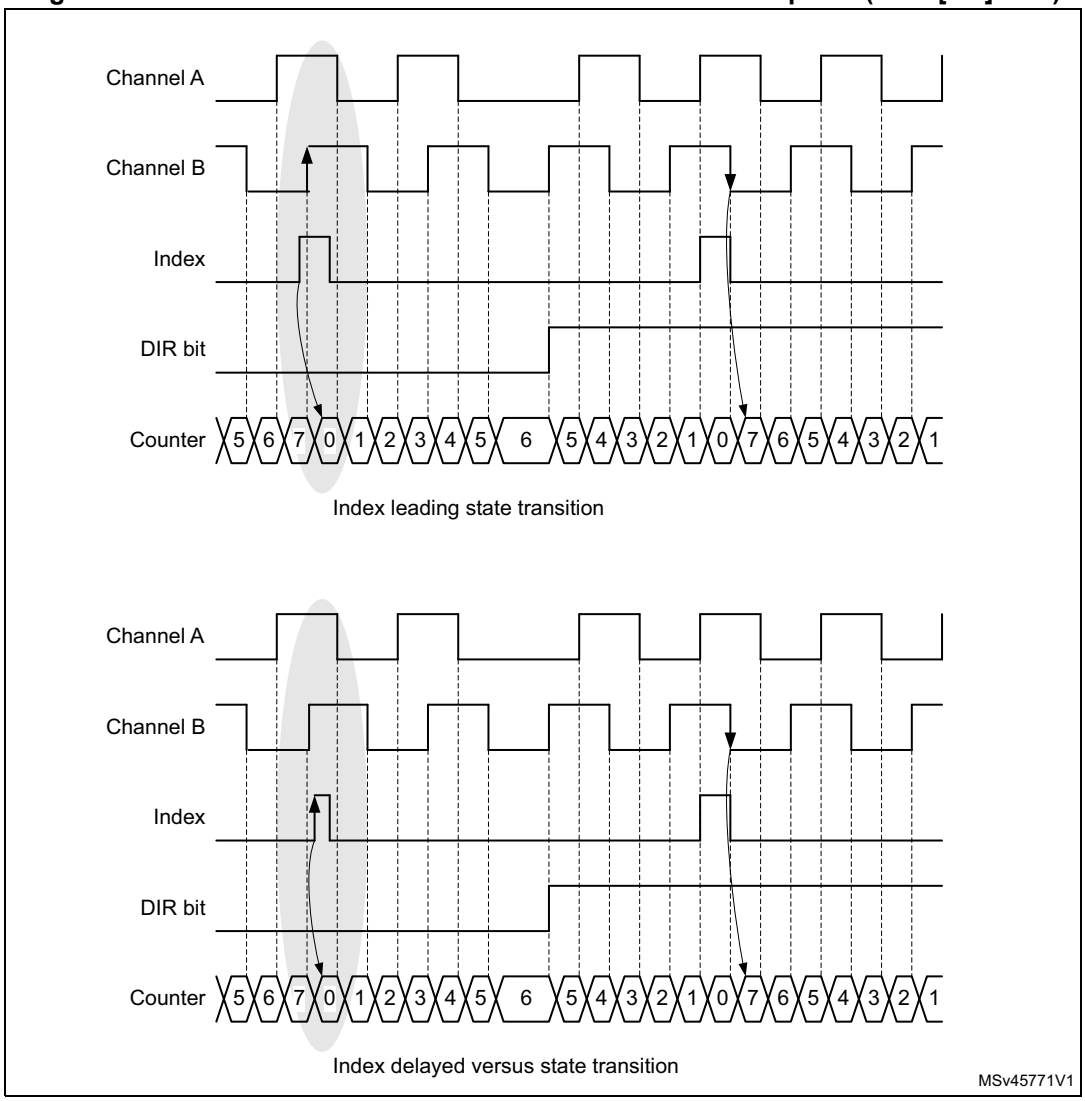


Figure 447. Counter reset Narrow index pulse (closer view, ARR = 0x07)

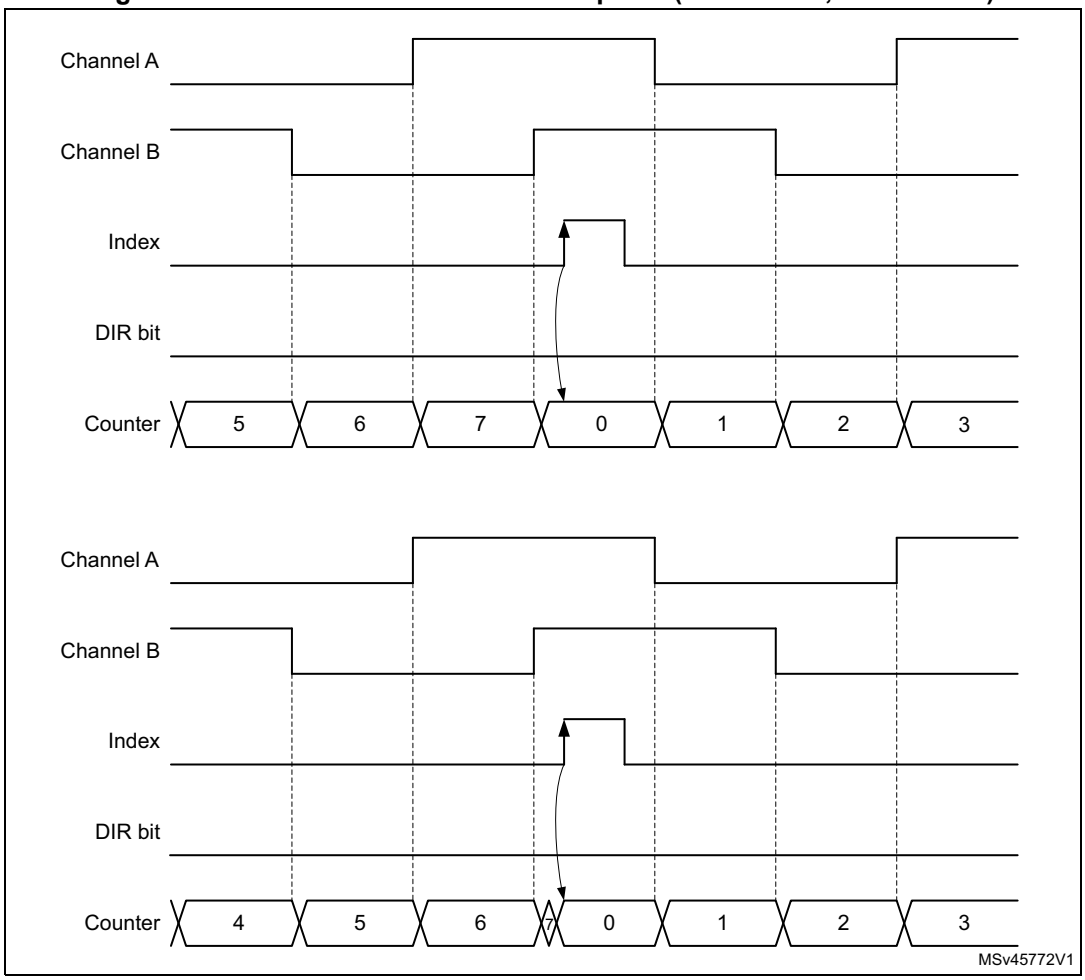
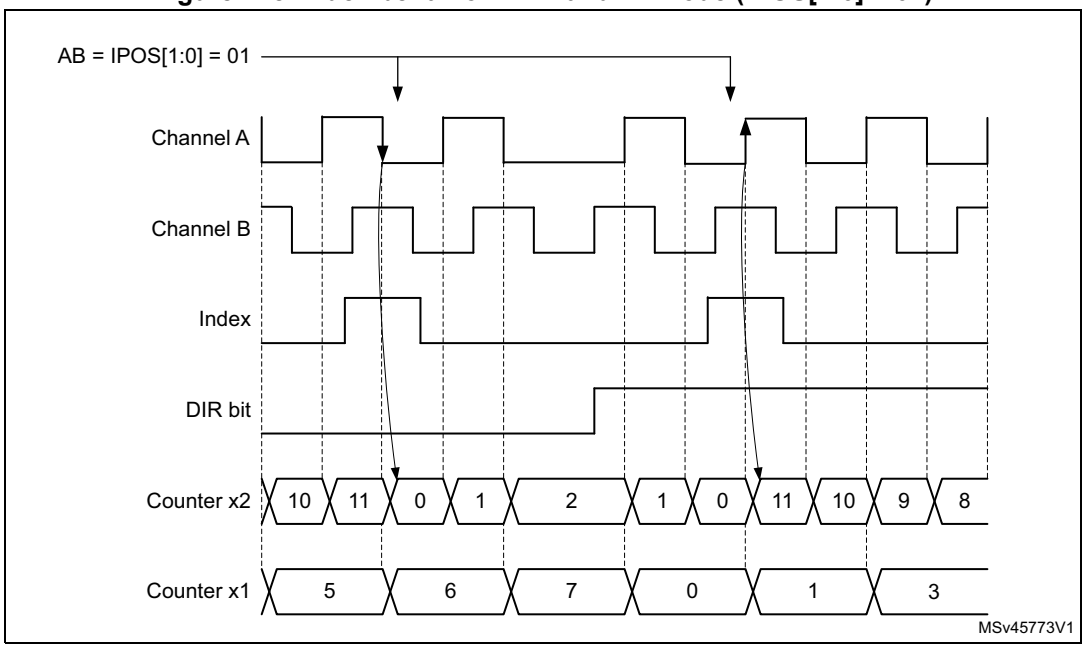


Figure 448 shows how the index is managed in x1 and x2 modes.

Figure 448. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

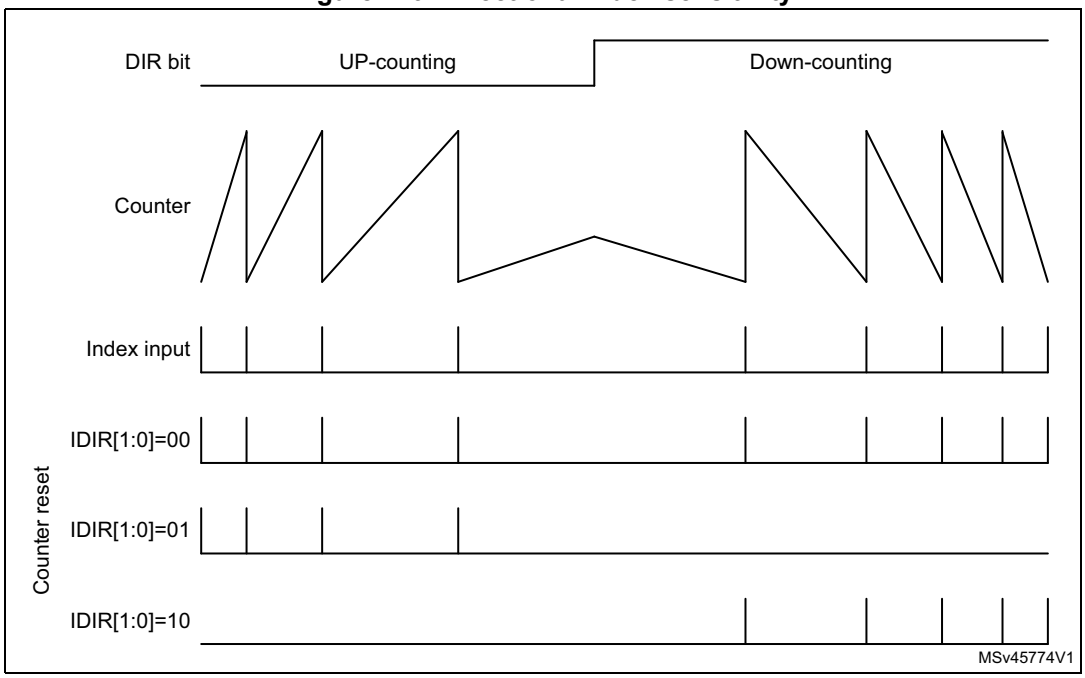


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 449 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

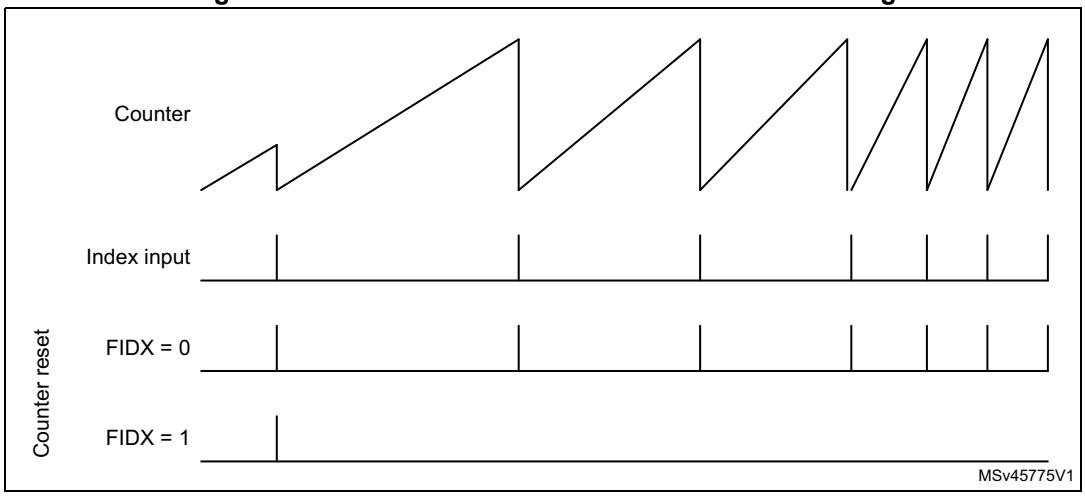
Figure 449. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 450*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 450. Counter reset as function of FIDX bit setting



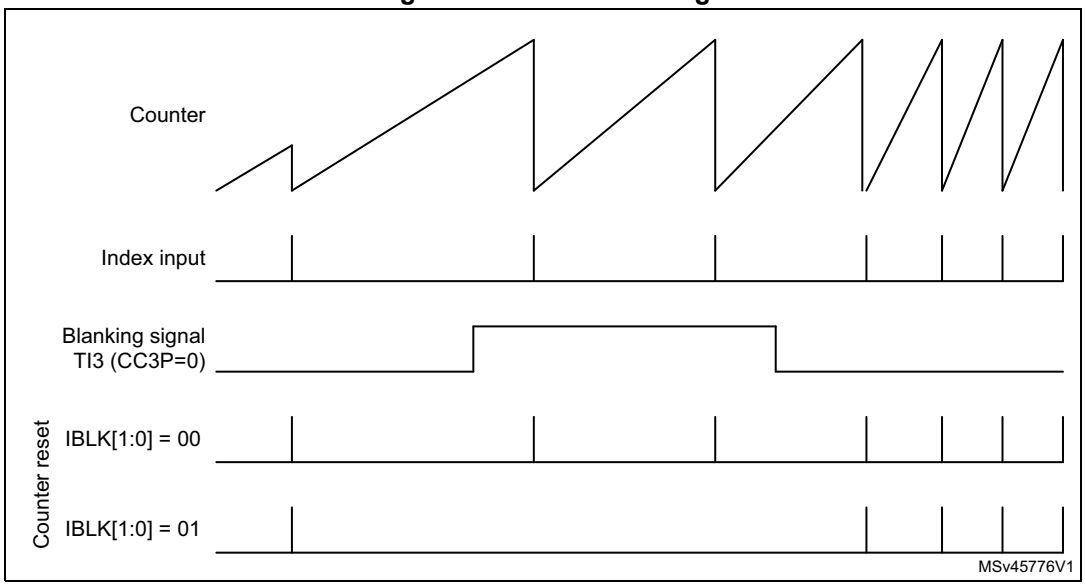
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 451*.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 451. Index blanking



Index management in nonquadrature mode

Figure 452 and *Figure 453* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 452. Index behavior in clock + direction mode, IPOS[0] = 1

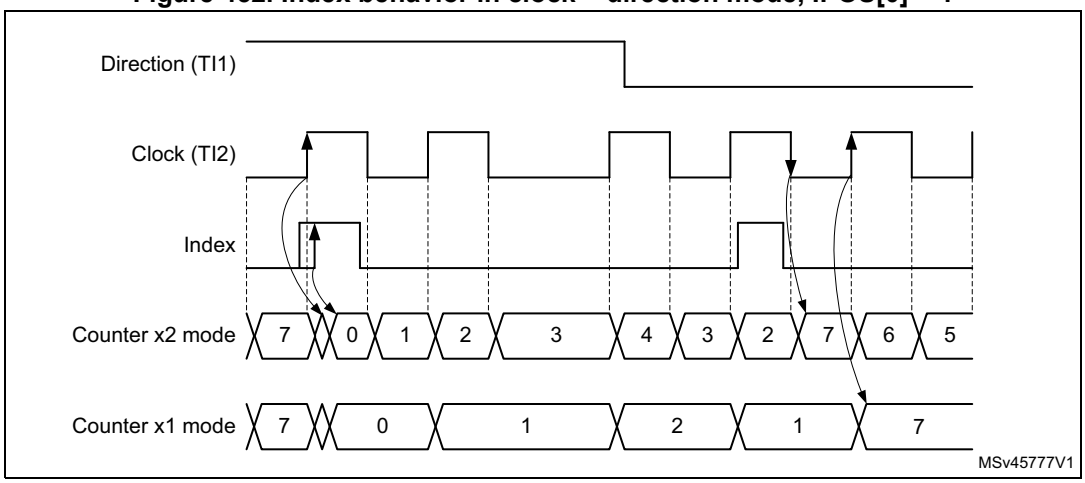
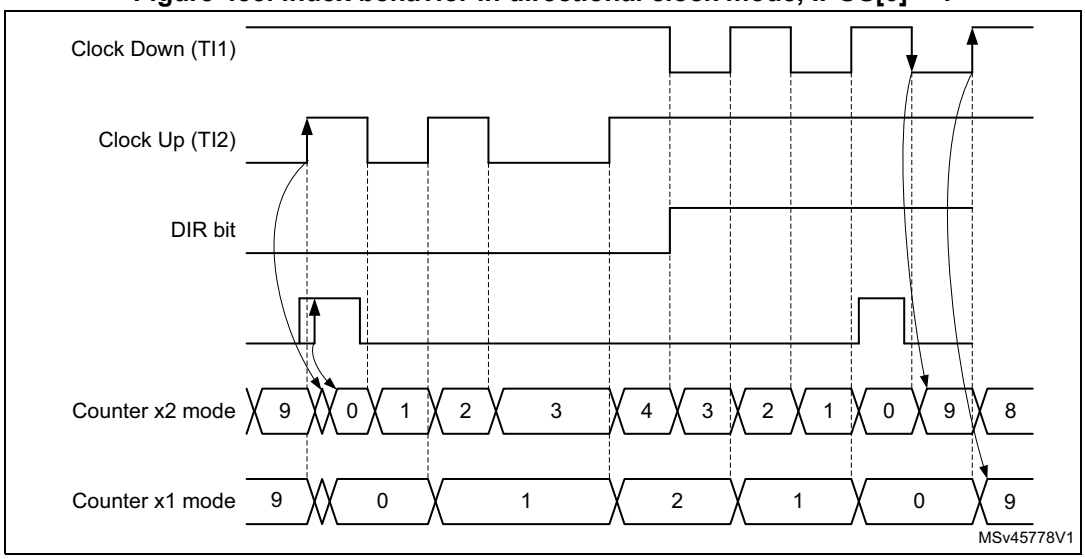


Figure 453. Index behavior in directional clock mode, IPOS[0] = 1

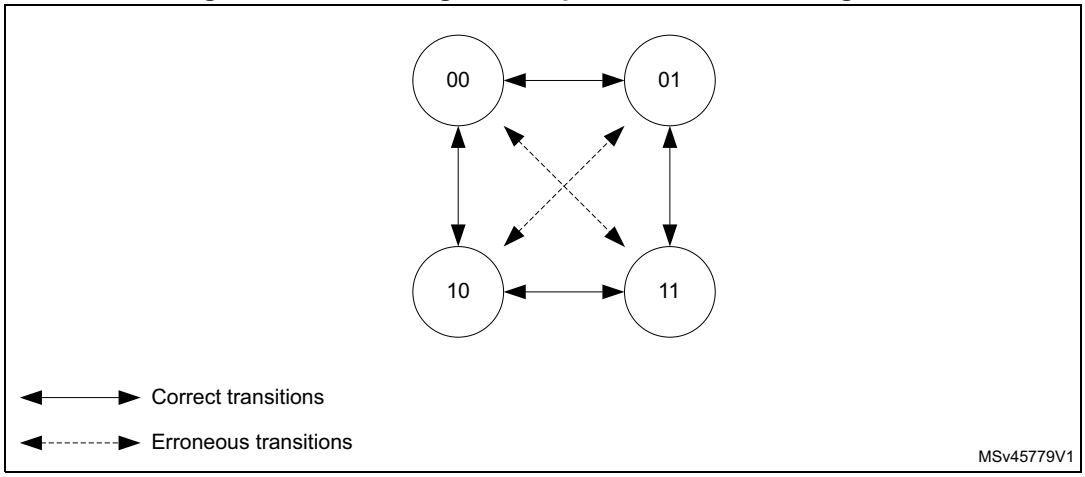


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 454*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 454. State diagram for quadrature encoded signals



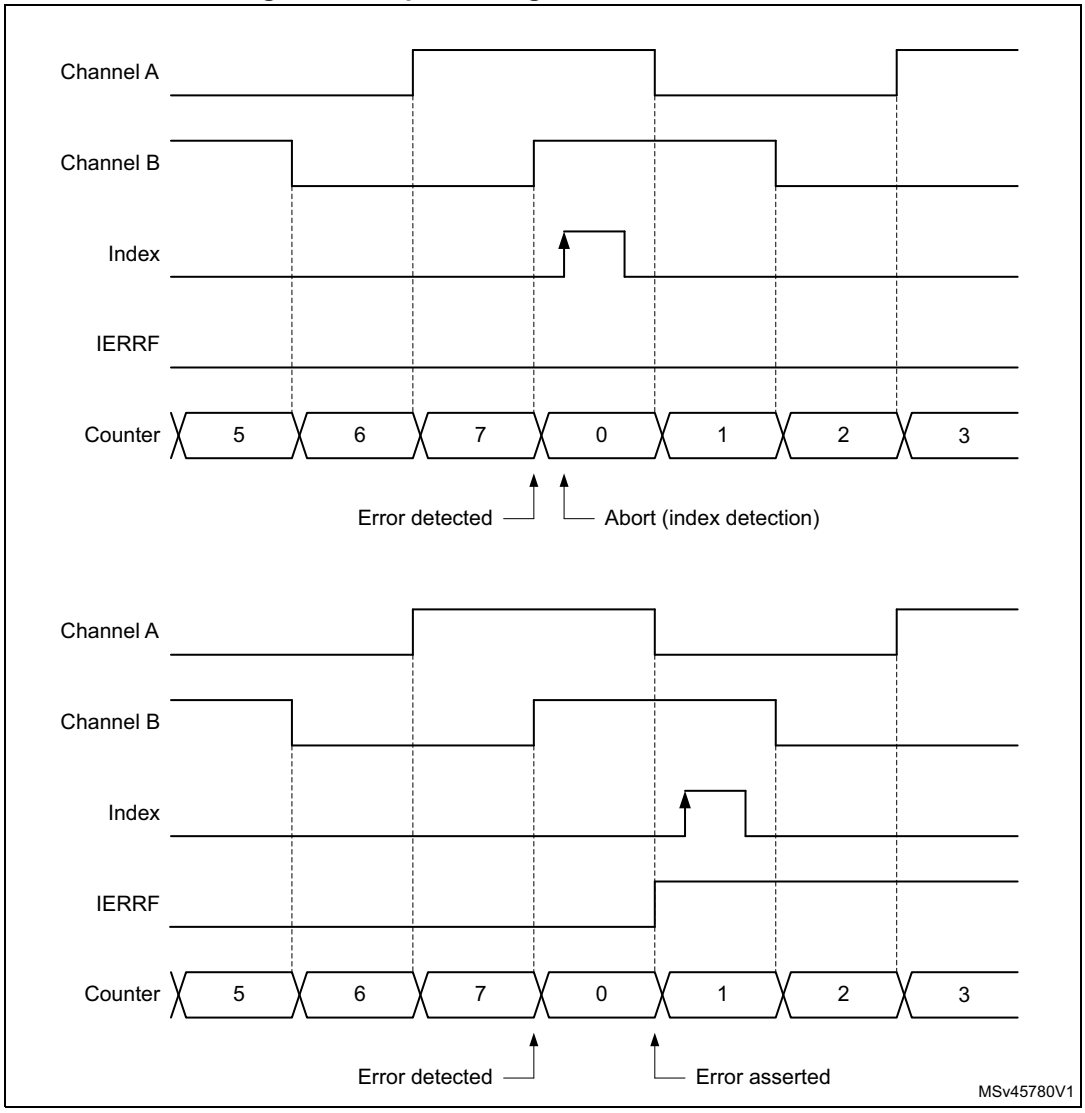
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

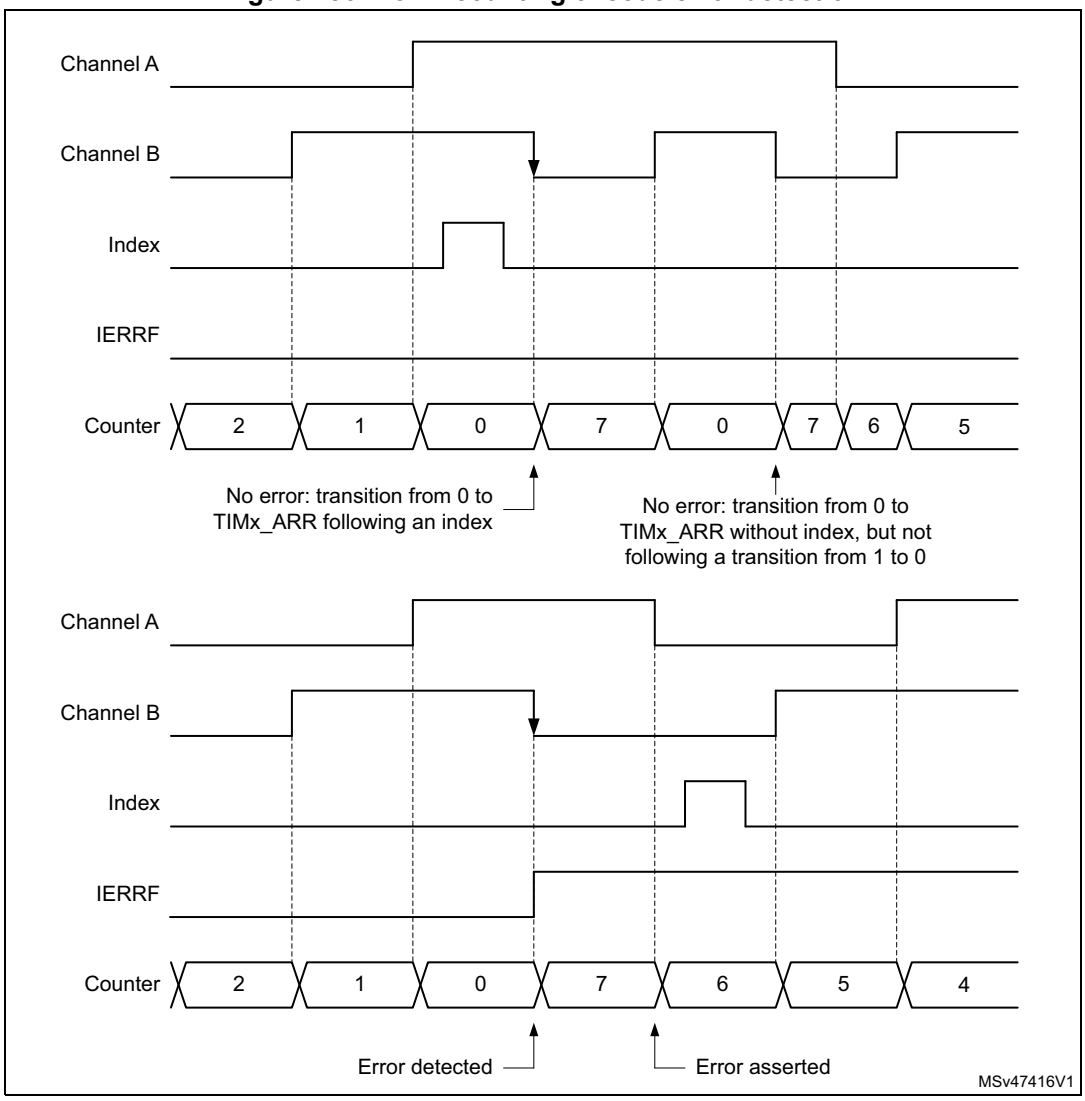
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 455*.

Figure 455. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 456*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 456. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

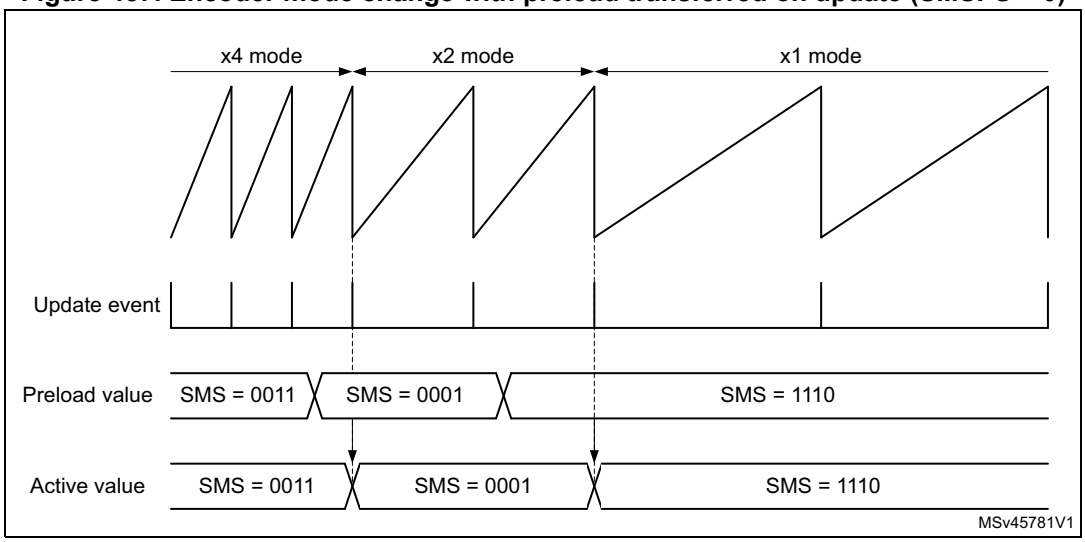
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 457*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 457. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

36.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

36.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

36.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 35.3.29: Interfacing with Hall sensors*.

36.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

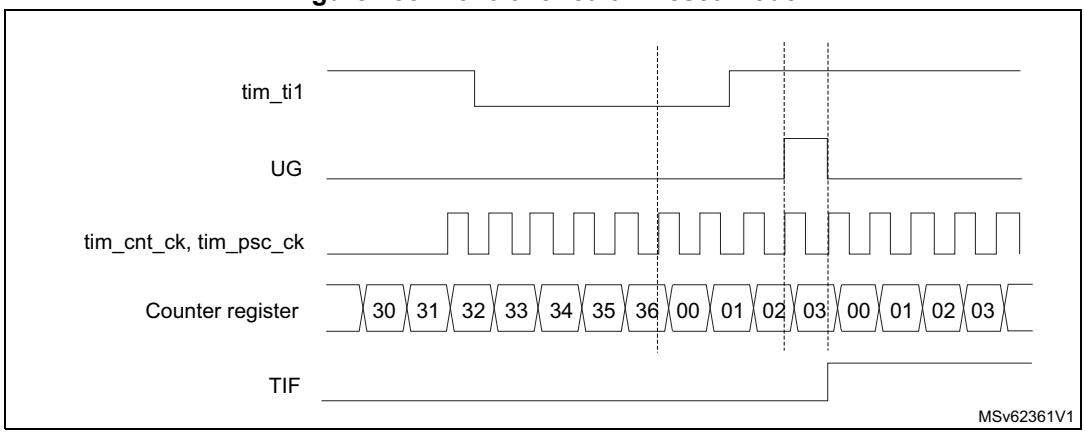
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the

trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 458. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

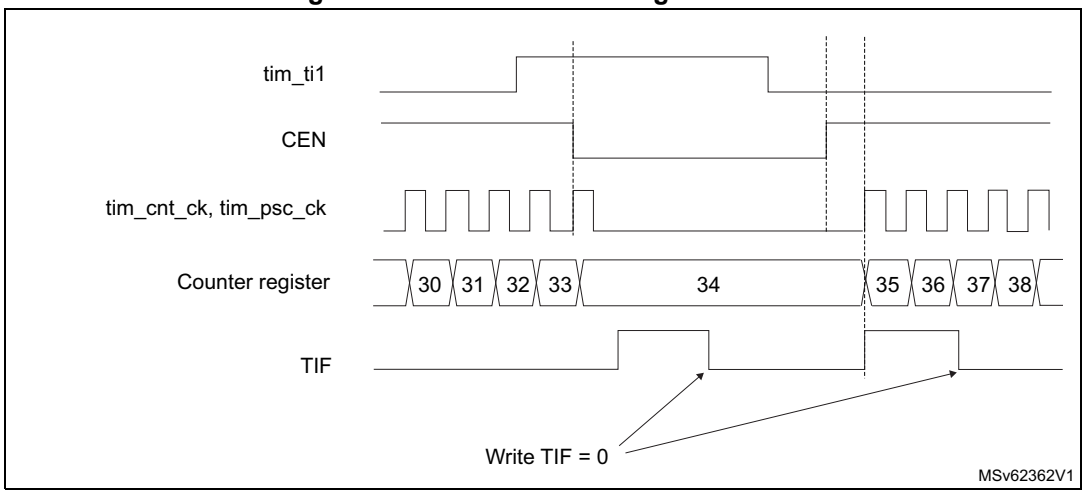
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 459. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

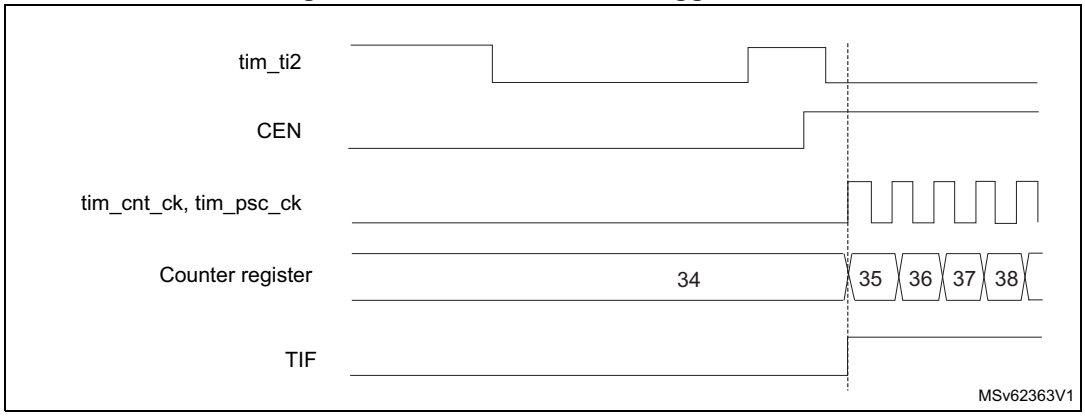
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 460. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

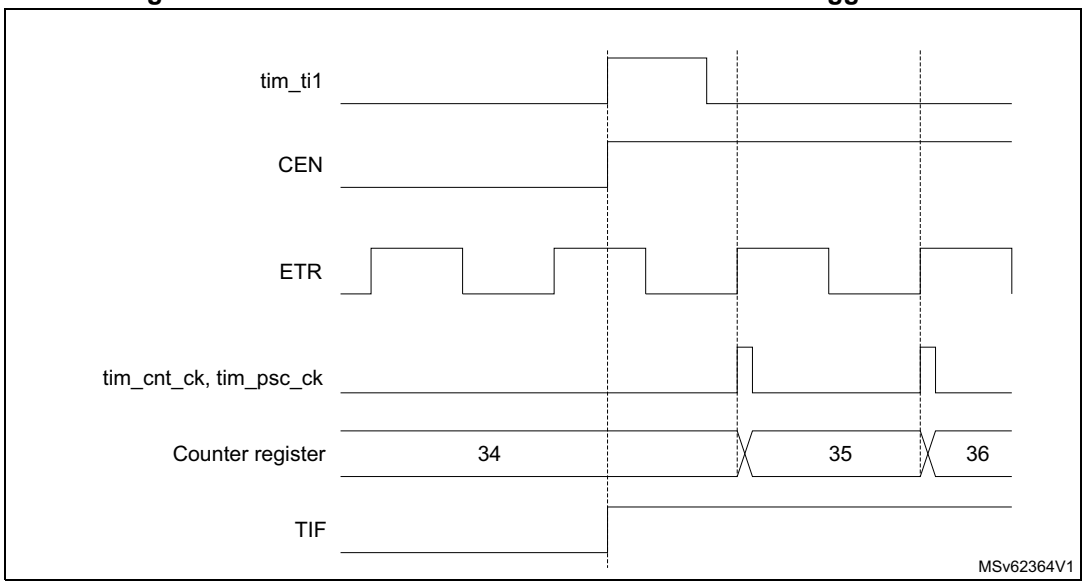
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 461. Control circuit in external clock mode 2 + trigger mode



36.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

Figure 462 and *Figure 463* show examples of master/slave timer connections.

Figure 462. Master/Slave timer example

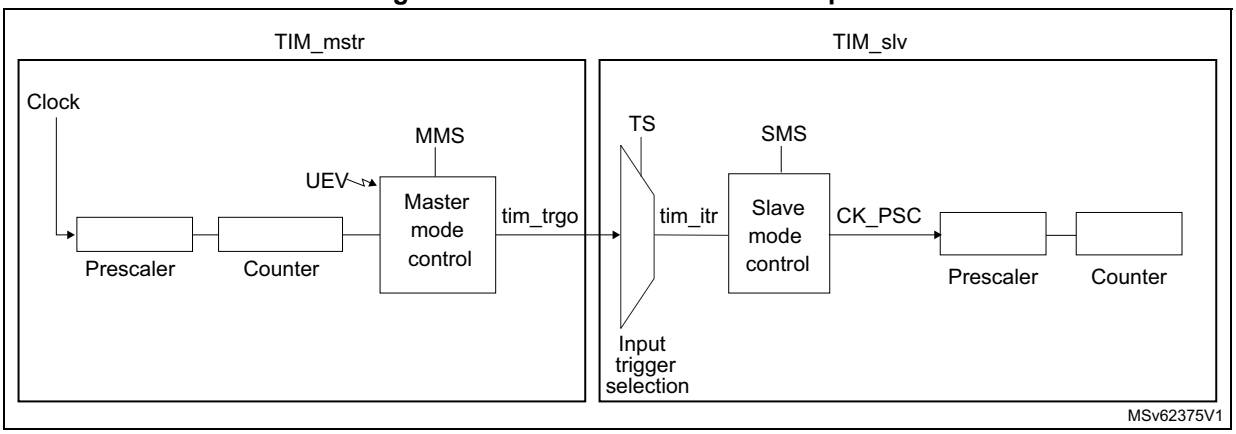
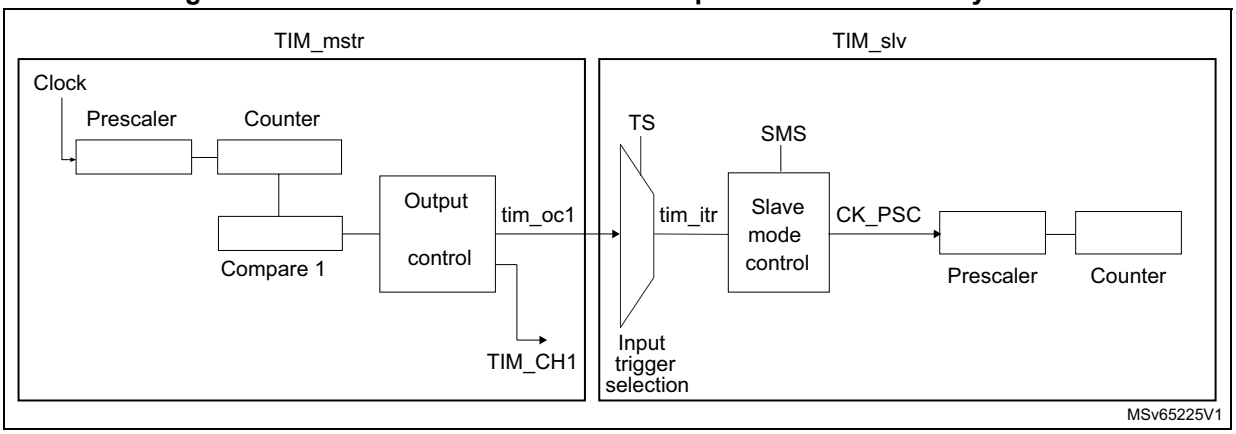


Figure 463. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 463) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 462*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

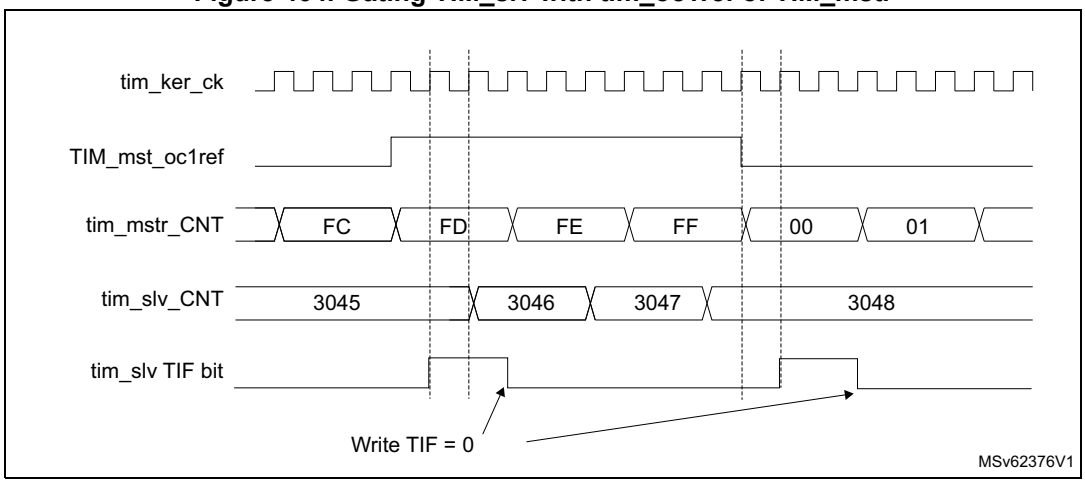
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 462* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 464. Gating TIM_slv with tim_oc1ref of TIM_mstr

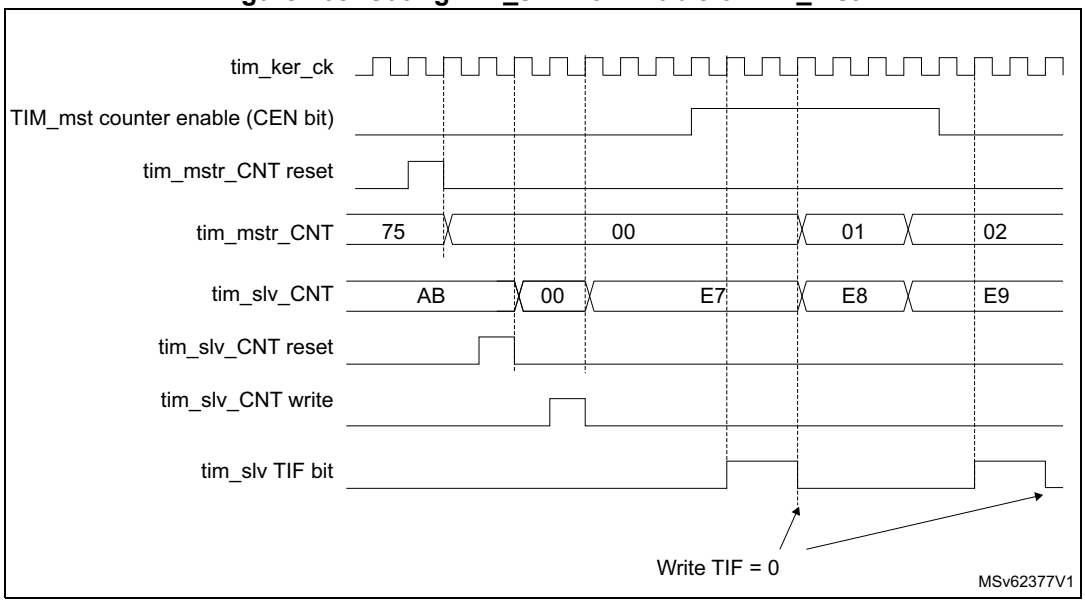


In the example in *Figure 464*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 465*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 465. Gating TIM_slv with Enable of TIM_mstr

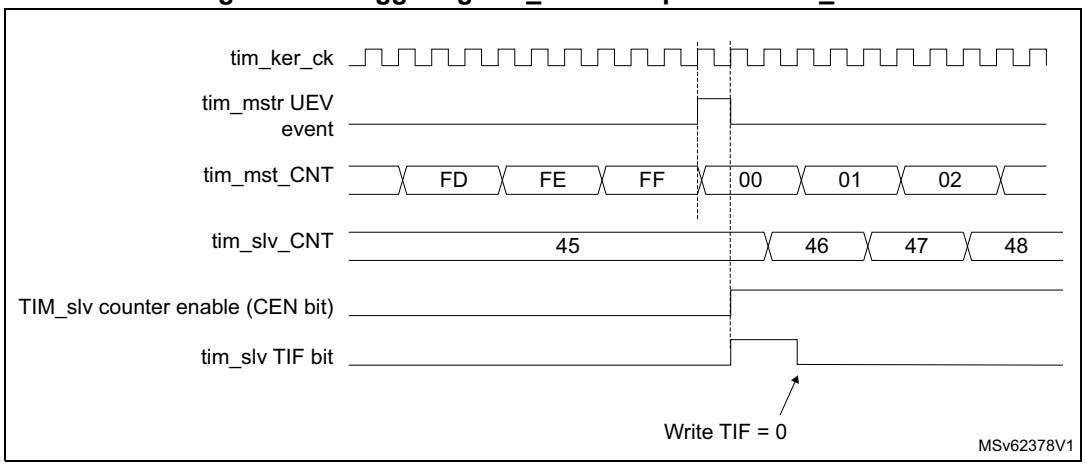


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 462* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

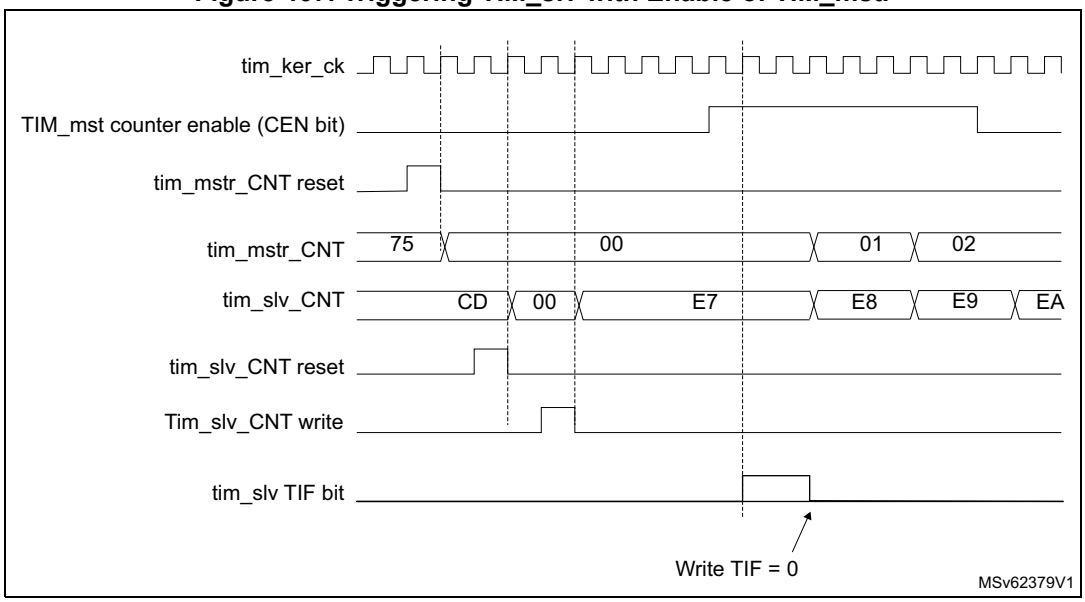
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 466. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 467* shows the behavior with the same configuration as in *Figure 466* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 467. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

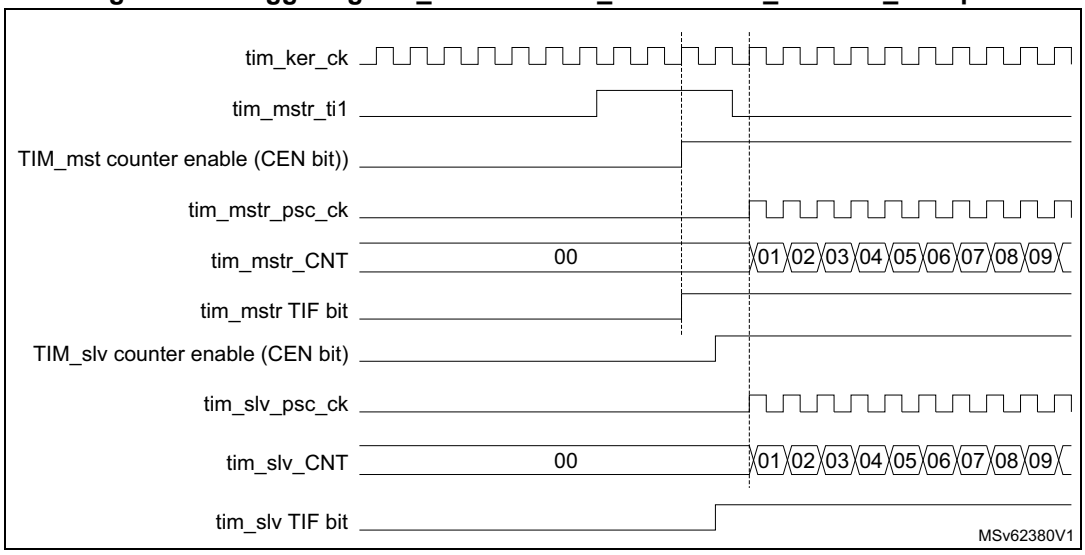
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 462* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 468. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.4.25 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

36.4.26 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 36.5.23: TIMx DMA control register (TIMx_DCR)(x = 2 to 4)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

36.4.27 TIM2/TIM3/TIM4 DMA requests

The TIM2/TIM3/TIM4 can generate a DMA requests, as shown in *Table 378*.

Table 378. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

36.4.28 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

36.4.29 TIM2/TIM3/TIM4 low-power modes

Table 379. Effect of low-power modes on TIM2/TIM3/TIM4

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

36.4.30 TIM2/TIM3/TIM4 interrupts

The TIM2/TIM3/TIM4 can generate multiple interrupts, as shown in *Table 380*.

Table 380. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

36.5 TIM2/TIM3/TIM4 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

36.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 4)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

36.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 4)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw			rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 36.4.25* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input

1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 35.3.29: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

36.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 4)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 36.3: TIM2/TIM3/TIM4 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001:Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010:Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011:Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111:External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010:Encoder mode: Clock plus direction, x2 mode.

1011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100:Encoder mode: Directional clock, x2 mode.

1101:Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

36.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 4)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

36.5.5 TIMx status register (TIMx_SR)(x = 2 to 4)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

36.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 4)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

36.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 4)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000:No filter, sampling is done at f_{DTS}

0001:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100:$f_{SAMPLING} = f_{DTS}/2$, N = 6

0101:$f_{SAMPLING} = f_{DTS}/2$, N = 8

0110:$f_{SAMPLING} = f_{DTS}/4$, N = 6

0111:$f_{SAMPLING} = f_{DTS}/4$, N = 8

1000:$f_{SAMPLING} = f_{DTS}/8$, N = 6

1001:$f_{SAMPLING} = f_{DTS}/8$, N = 8

1010:$f_{SAMPLING} = f_{DTS}/16$, N = 5

1011:$f_{SAMPLING} = f_{DTS}/16$, N = 6

1100:$f_{SAMPLING} = f_{DTS}/16$, N = 8

1101:$f_{SAMPLING} = f_{DTS}/32$, N = 5

1110:$f_{SAMPLING} = f_{DTS}/32$, N = 6

1111:$f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

36.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 4)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

36.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 4)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

36.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 4)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000:Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001:Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010:Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011:Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100:Force inactive level - tim_oc3ref is forced low.

0101:Force active level - tim_oc3ref is forced high.

0110:PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111:PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010:Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011:Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100:Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110:Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111:Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

36.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 4)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1:non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 381. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

36.5.12 TIMx counter (TIMx_CNT)(x = 2 to 4)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

36.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 4)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

36.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 4)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 36.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

36.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 4)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

36.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 4)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

36.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

36.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

36.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 4)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

36.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 4)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

36.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 4)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

36.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 4)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 36.4.2: TIM2/TIM3/TIM4 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

36.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 4)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

36.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 4)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

36.5.25 TIMx register map

TIMx registers are mapped as described in the table below.

Table 382. TIM2/TIM3/TIM4 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value				0			0																		0	0	0	0	0			
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value								0	0	0	0							0		0	0	0	0	0		0		0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 382. TIM2/TIM3/TIM4 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	TIMx_CNT	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR (x = 2 to 4)	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved	Reserved																															
0x034	TIMx_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0

Table 382. TIM2/TIM3/TIM4 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 Basic timers (TIM6/TIM7)

37.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

37.2 TIM6/TIM7 main features

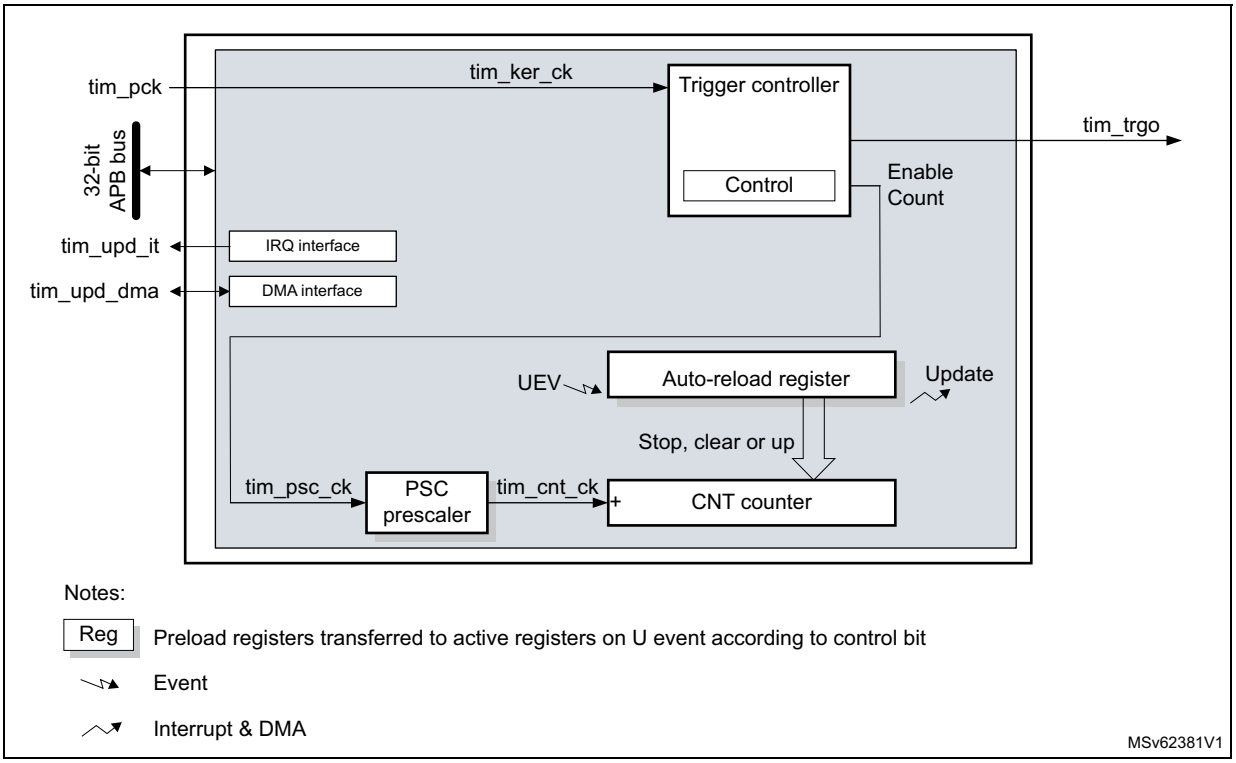
Basic timer (TIM6/TIM7) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.
- ADC synchronization for jitter-free sampling points.

37.3 TIM6/TIM7 functional description

37.3.1 TIM6/TIM7 block diagram

Figure 469. Basic timer block diagram



37.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 383. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

37.3.3 TIM6/TIM7 clocks

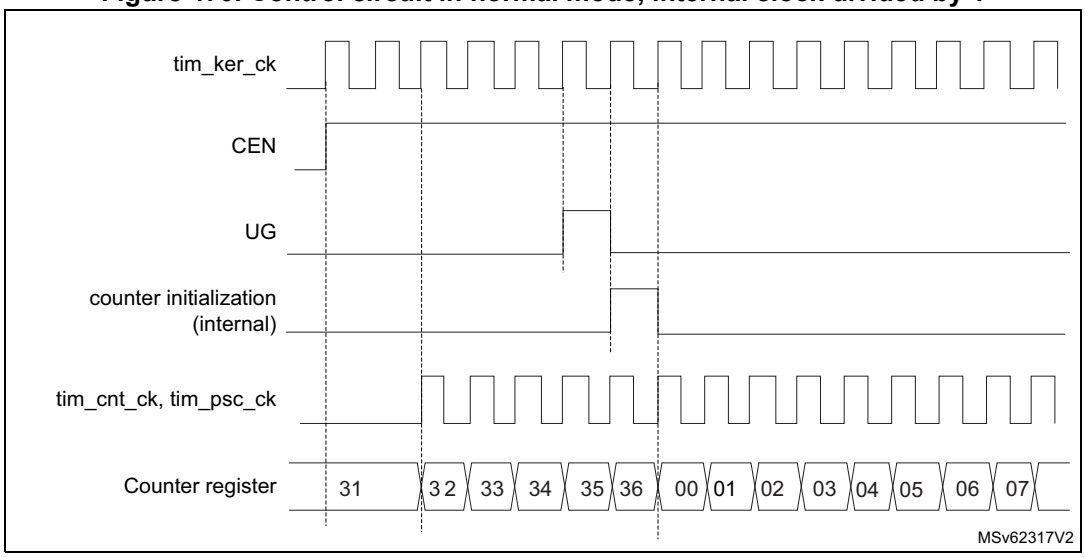
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 470 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 470. Control circuit in normal mode, internal clock divided by 1



37.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 471 and *Figure 472* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 471. Counter timing diagram with prescaler division change from 1 to 2

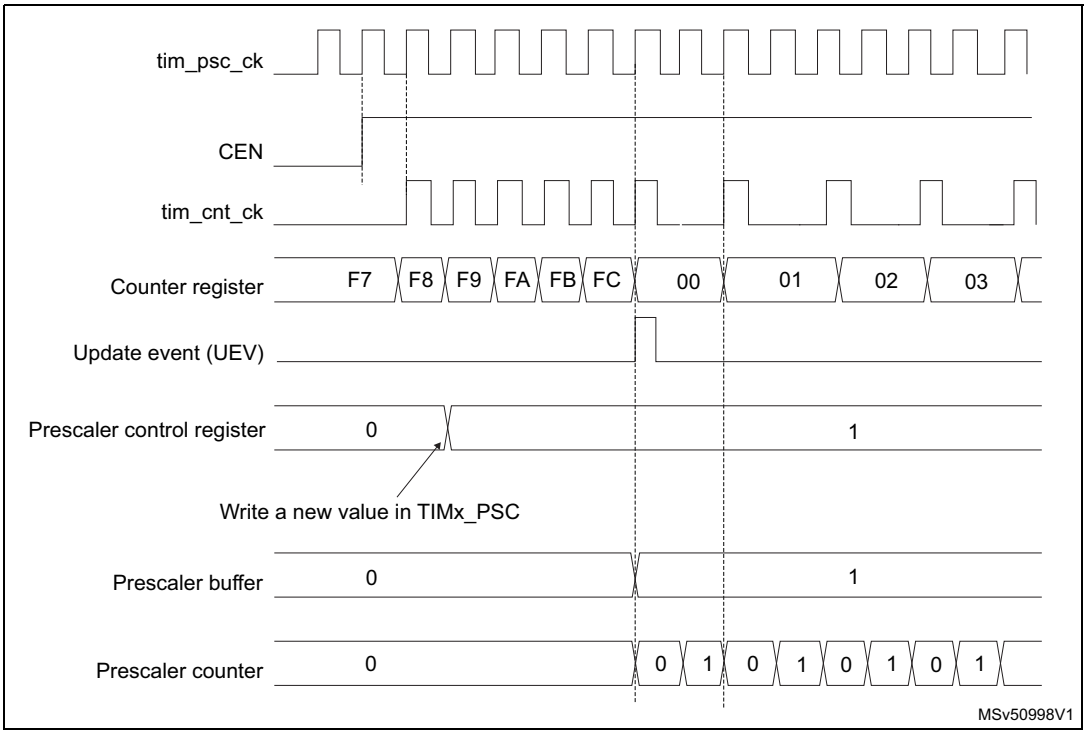
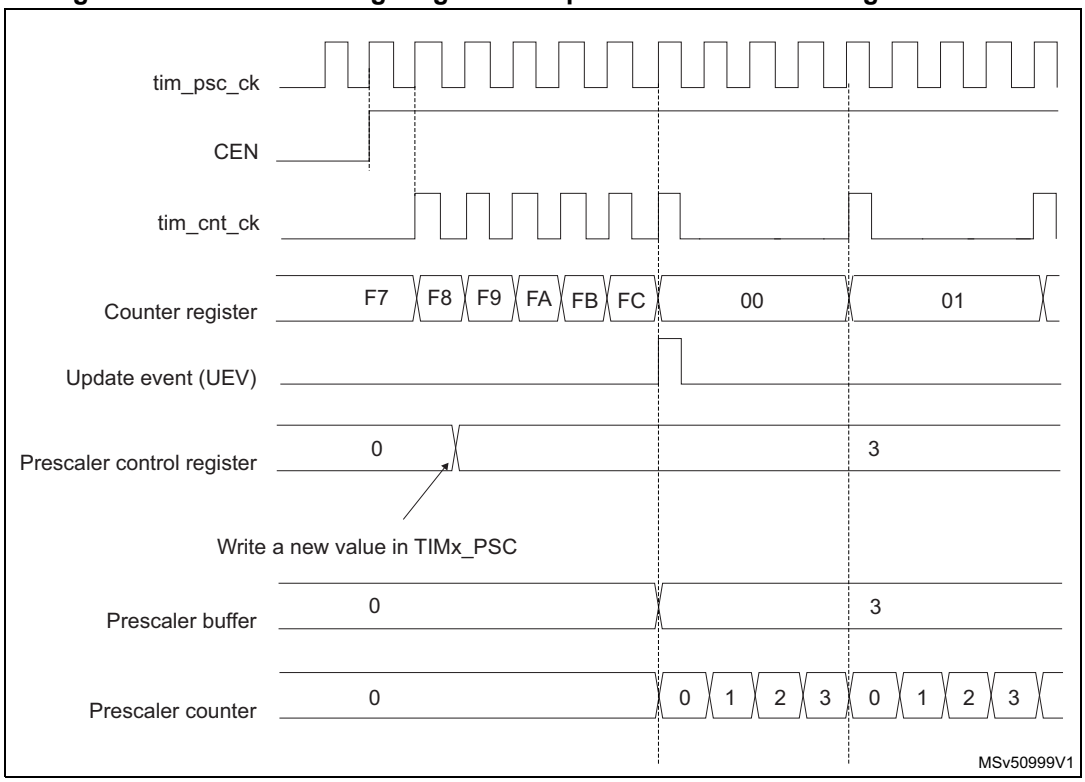


Figure 472. Counter timing diagram with prescaler division change from 1 to 4



37.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 473. Counter timing diagram, internal clock divided by 1

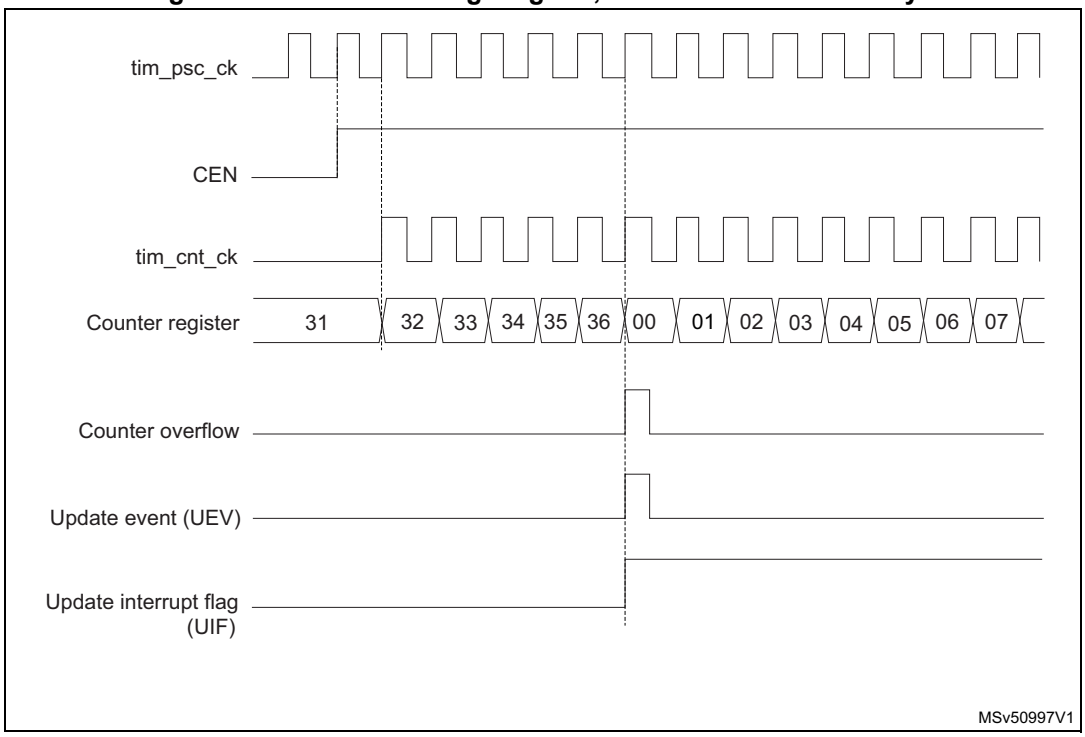


Figure 474. Counter timing diagram, internal clock divided by 2

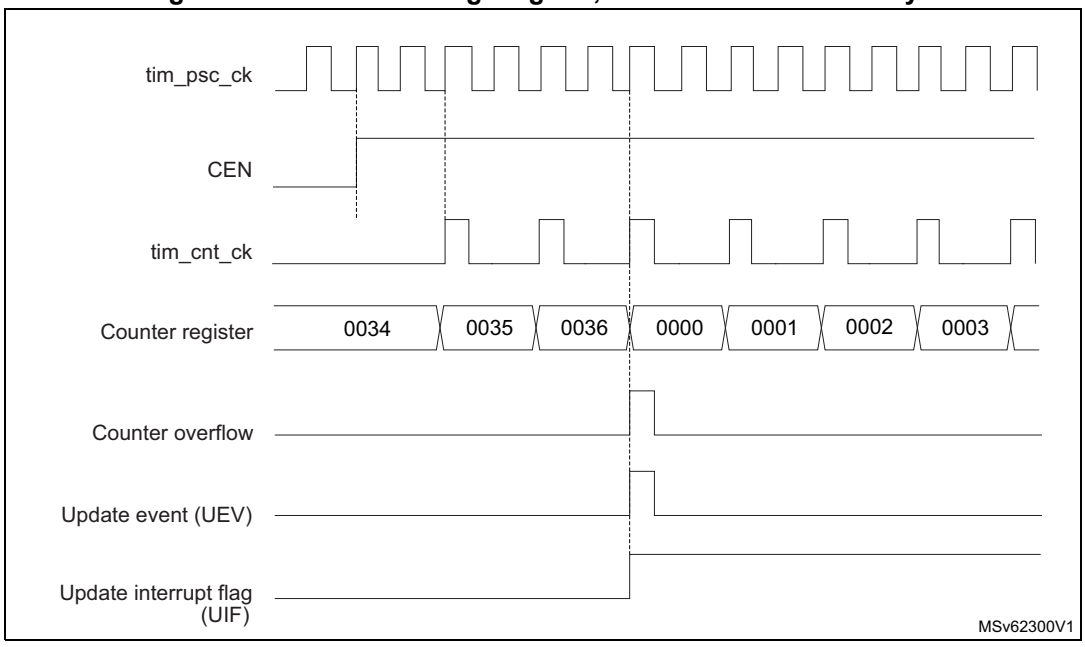


Figure 475. Counter timing diagram, internal clock divided by 4

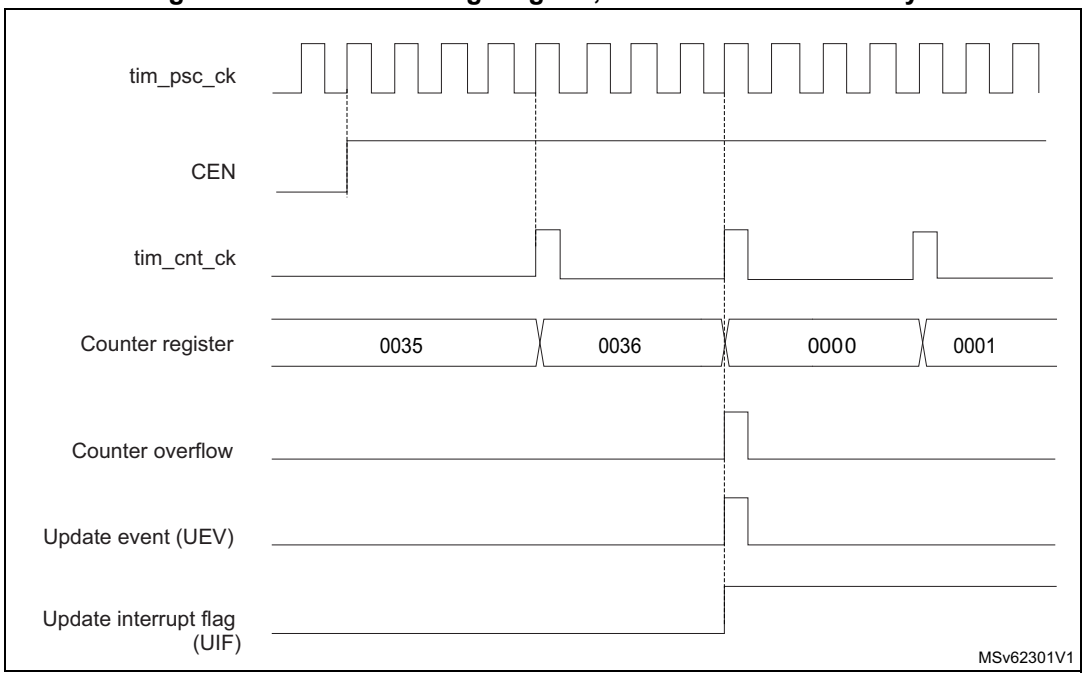


Figure 476. Counter timing diagram, internal clock divided by N

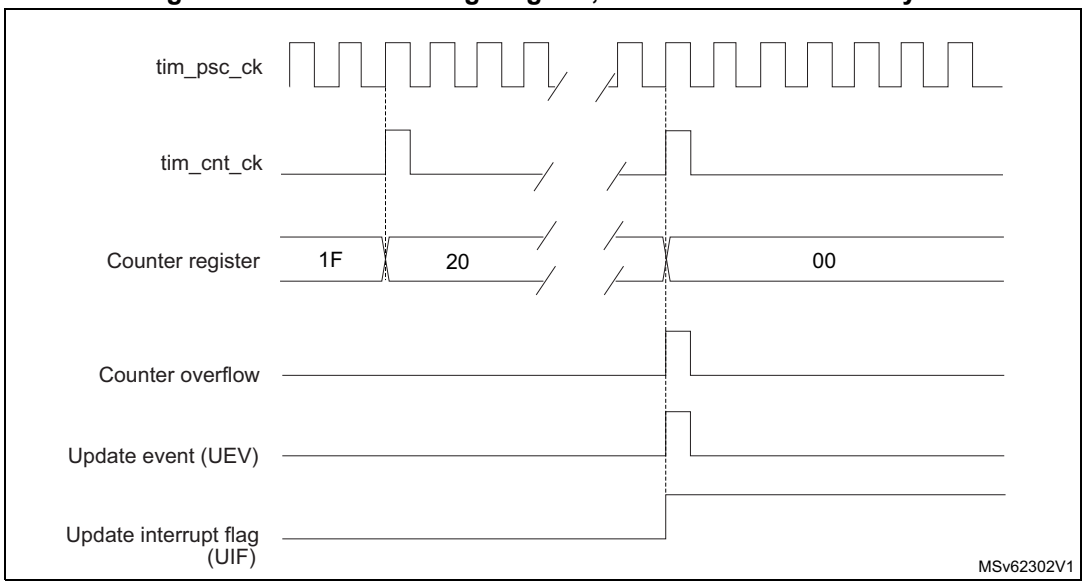


Figure 477. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

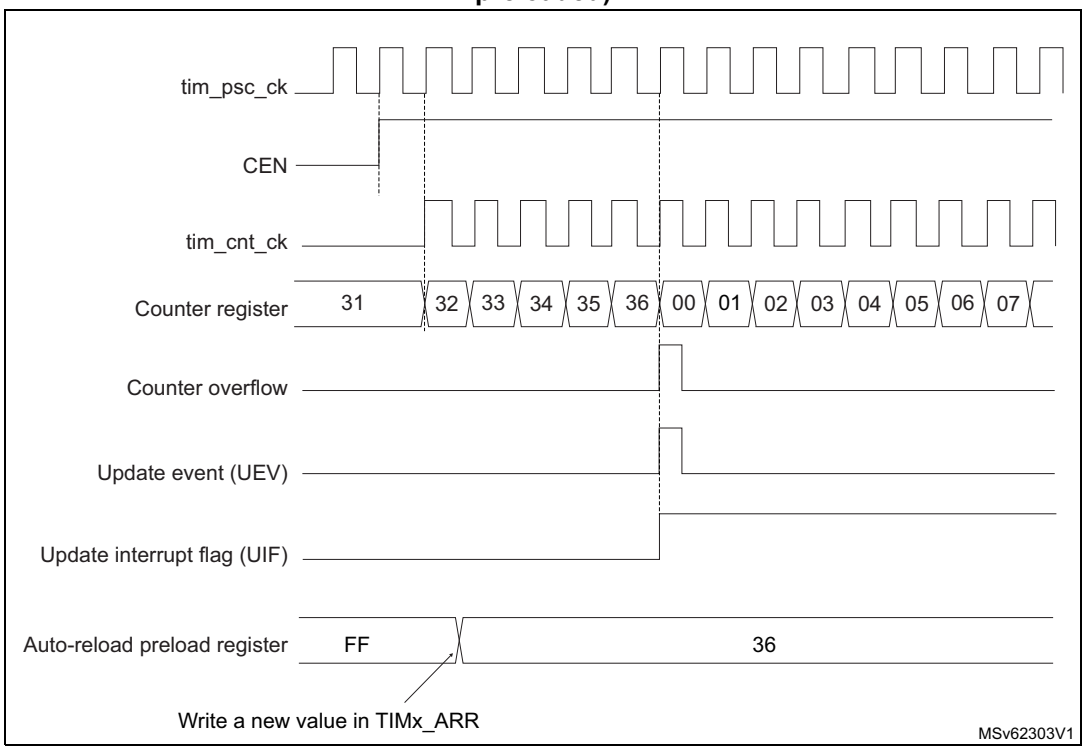
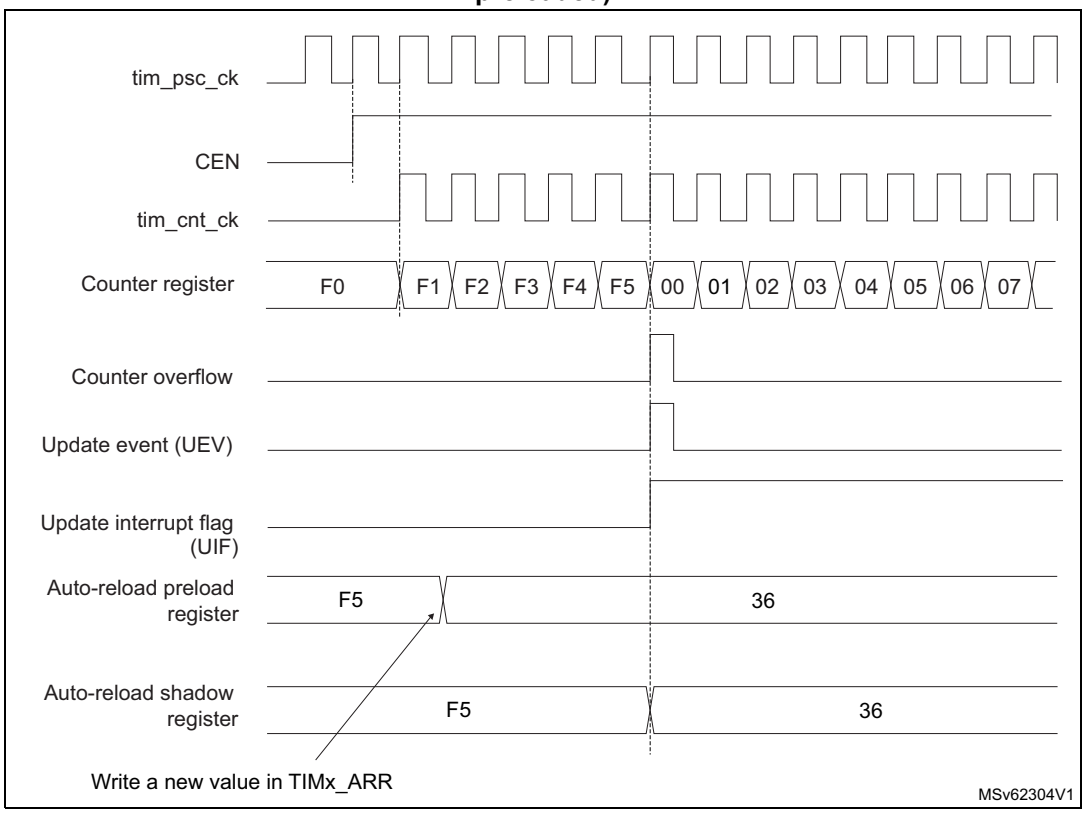


Figure 478. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



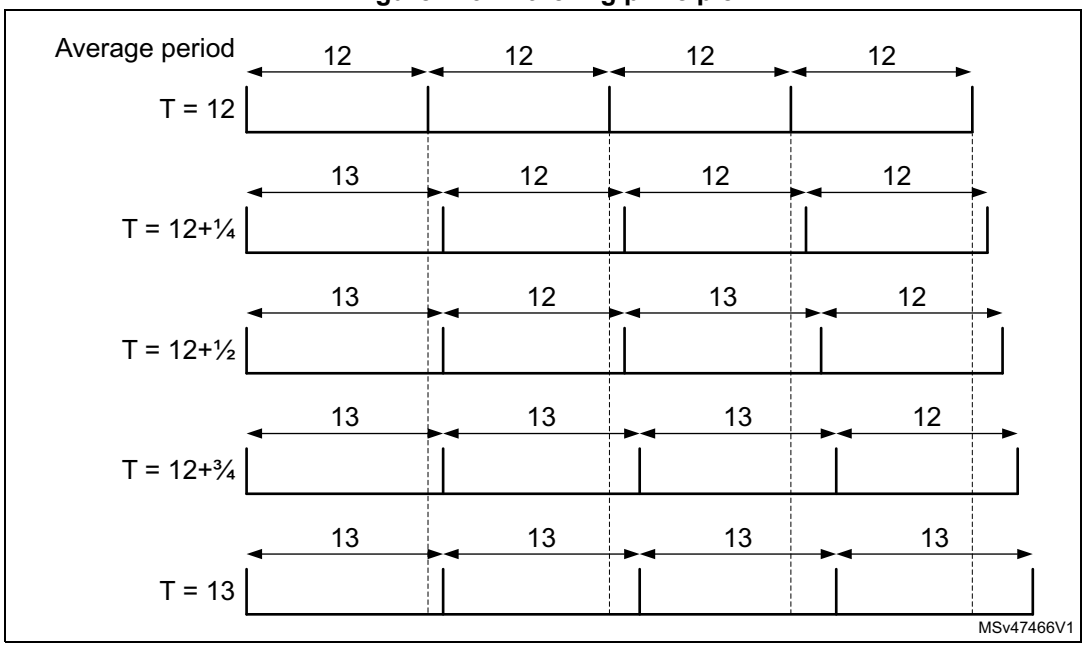
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 479 presents the dithering principle applied to four consecutive counting periods.

Figure 479. Dithering principle

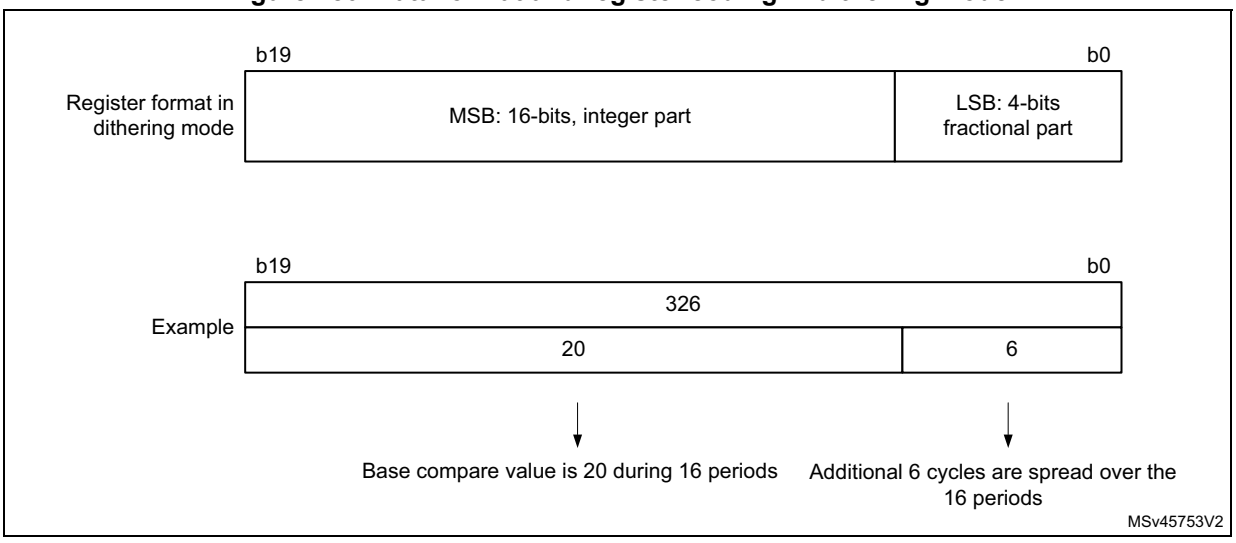


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 480* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 480. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

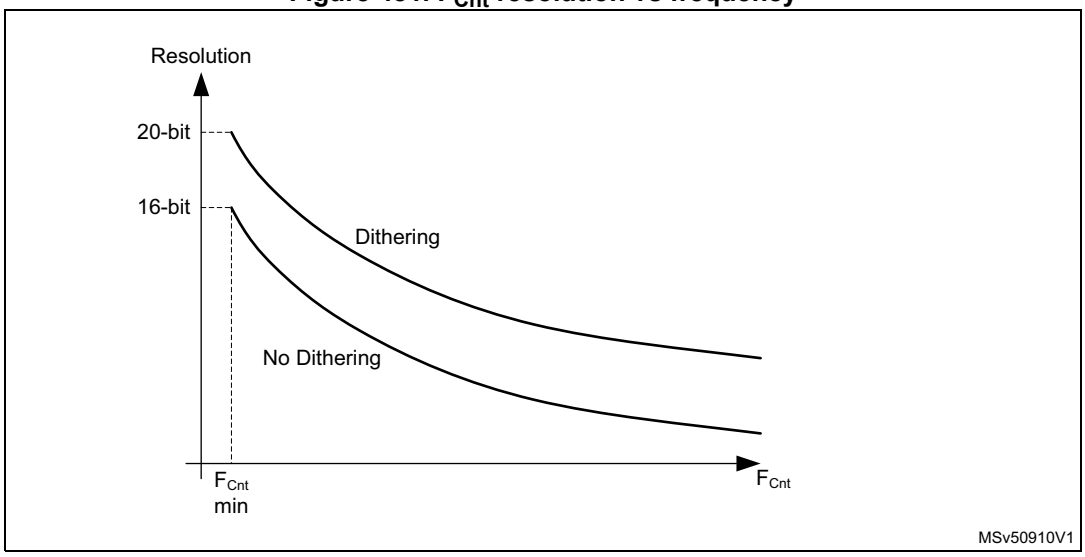
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

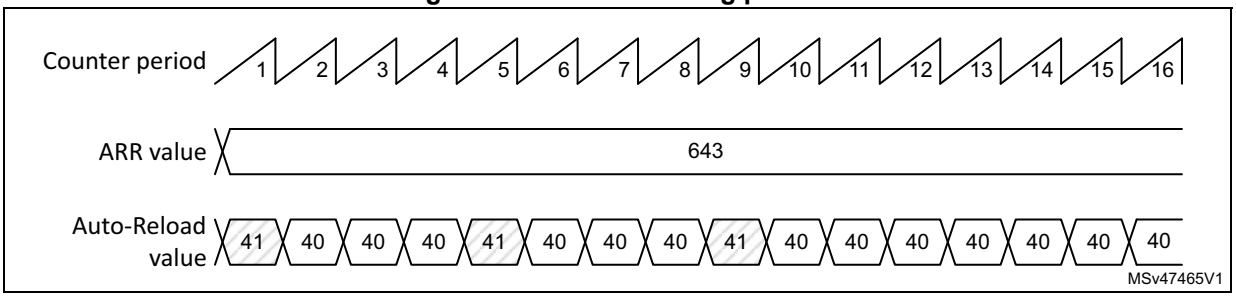
As shown on *Figure 481*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 481. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 482*.

Figure 482. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 384*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 384. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

37.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

37.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

37.3.8 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

37.3.9 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 385*.

Table 385. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

37.3.10 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

37.3.11 TIM6/TIM7 low-power modes

Table 386. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

37.3.12 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 387*.

Table 387. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

37.4 TIM6/TIM7 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

37.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

37.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 37.3.8* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

37.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

37.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

37.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

37.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

37.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

37.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 37.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

37.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 388. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value				0																						0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3* for the register boundary addresses.

38 General purpose timers (TIM15/TIM16/TIM17)

38.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 38.4.26: Timer synchronization (TIM15 only)*.

38.2 TIM15 main features

TIM15 includes the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- ADC synchronization for jitter-free sampling points

38.3 TIM16/TIM17 main features

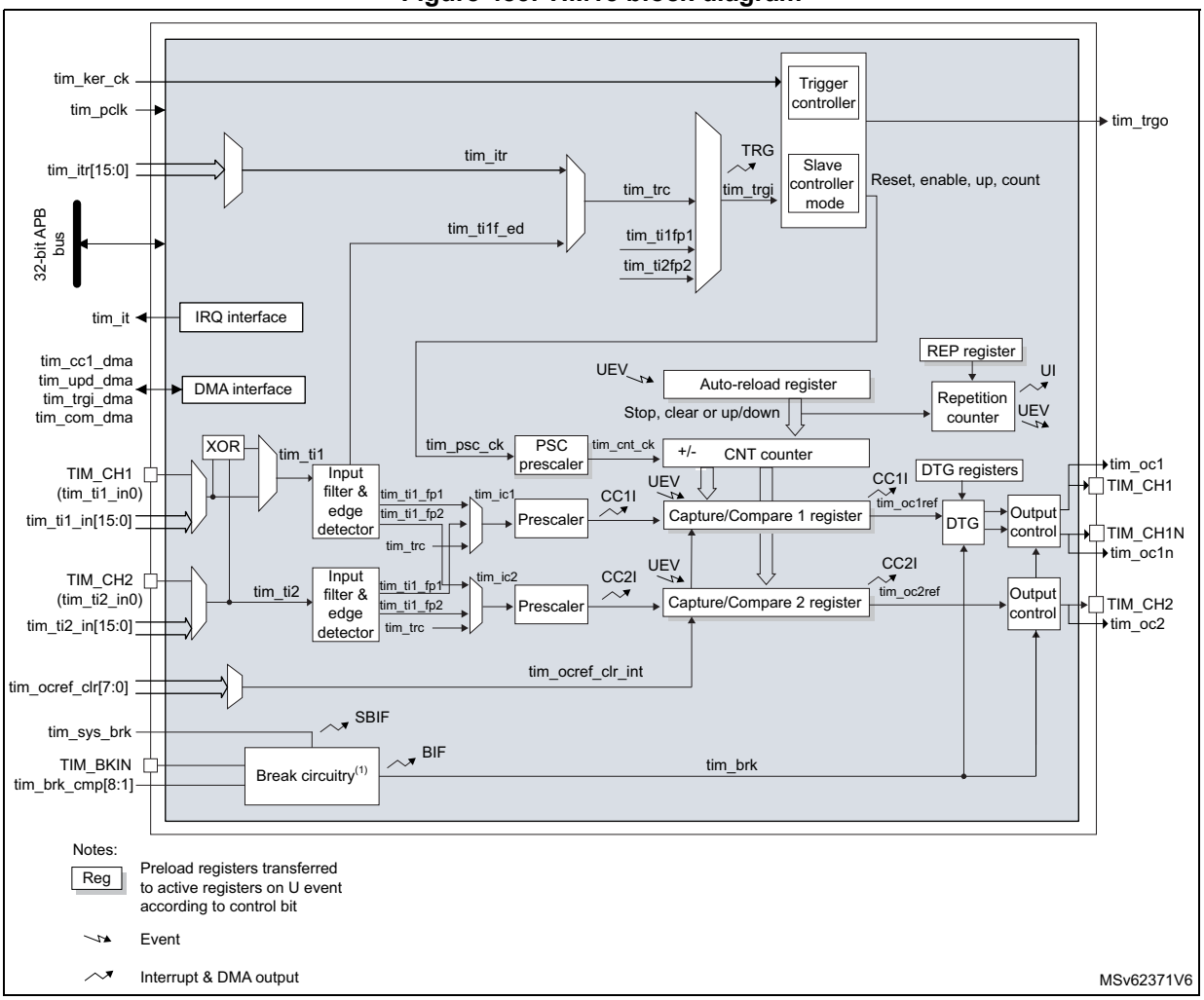
The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

38.4 TIM15/TIM16/TIM17 functional description

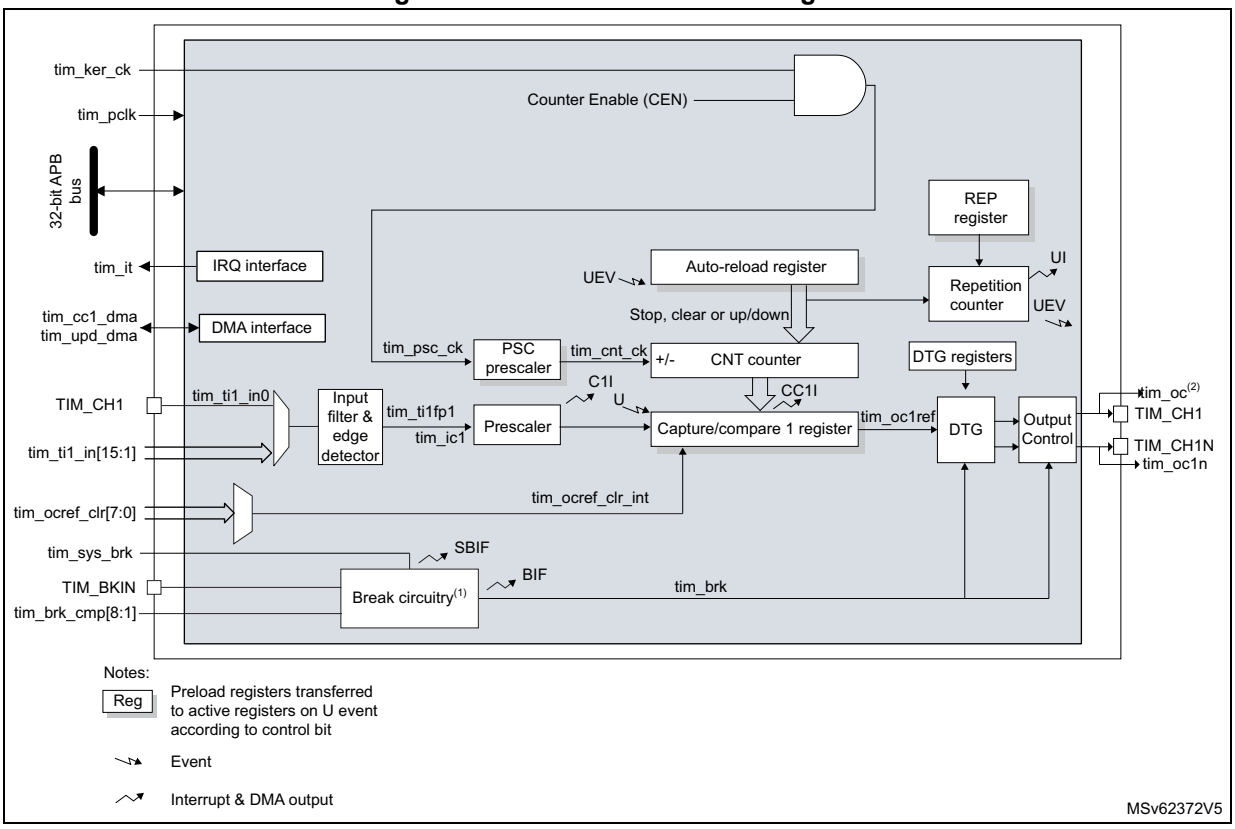
38.4.1 Block diagram

Figure 483. TIM15 block diagram



1. Refer to *Section 38.4.15: Using the break function* for details.

Figure 484. TIM16/TIM17 block diagram



1. Refer to *Section 38.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 38.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

38.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 389 and *Table 390* in this section summarize the TIM inputs and outputs.

Table 389. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 390. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Tables below list the sources connected to the tim_ti[2:1] input multiplexers.

Table 391. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	LSE	Reserved	
tim_ti1_in2	comp1_out	MCO	
tim_ti1_in3	comp2_out	HSE / 32	
tim_ti1_in4	i3c1_ibiack	RTC Wake-up	

Table 391. Interconnect to the tim_ti1 input multiplexer (continued)

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in5	Reserved	LSE	
tim_ti1_in6		LSI	
tim_ti1_in7		MSIS / 1024	
tim_ti1_in8		MSIS / 4	
tim_ti1_in9		HSI / 256	
tim_ti1_in10		i3c1_ibiack	i3c2_ibiack
tim_ti1_in[15:11]	Reserved		

Table 392. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	TIM15 sources
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	comp2_out
tim_ti2_in2	i3c2_ibiack
tim_ti2_in[15:3]	Reserved

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 393. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	Reserved
tim_itr5	Reserved
tim_itr6	Reserved
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr[15:9]	Reserved

Tables below list the sources connected to the tim_brk and tim_brk2inputs.

Table 394. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp[8:3]	Reserved		

Table 395. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SYSCFG_CFGR2
tim_sys_brk0	Cortex®-M33LOCKUP	CLL
tim_sys_brk1	Programmable voltage detector (PVD)	PVDL
tim_sys_brk2	SRAM2 parity error	SPL
tim_sys_brk3	Flash memory double ECC error	ECCL
tim_sys_brk4	Clock security system (CSS)	None (always enabled)

The table below lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 396. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signal assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr[7:2]	Reserved		

38.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload

preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Table 495 and *Table 486* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 485. Counter timing diagram with prescaler division change from 1 to 2

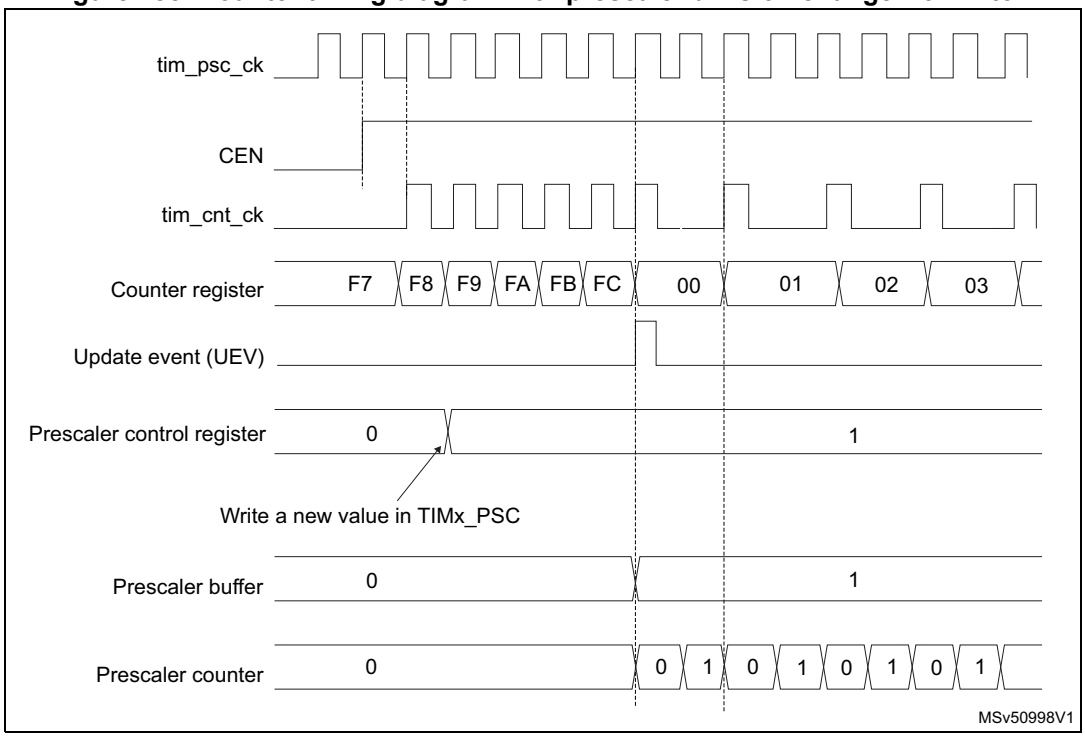
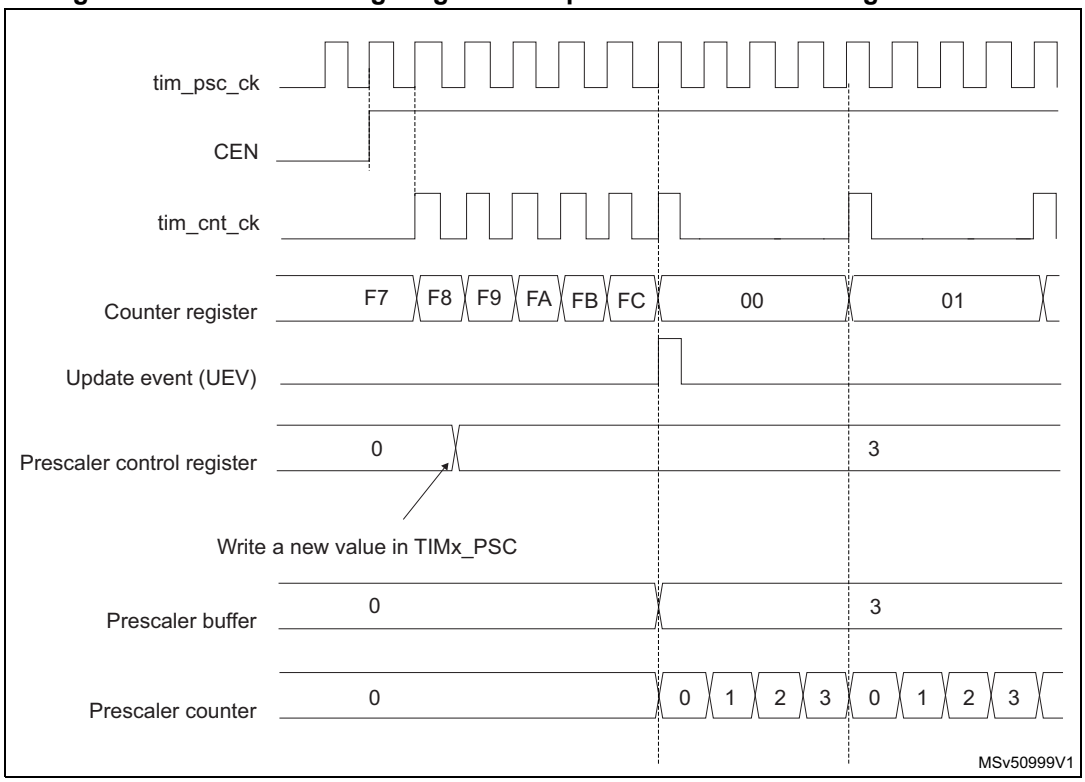


Figure 486. Counter timing diagram with prescaler division change from 1 to 4



38.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 487. Counter timing diagram, internal clock divided by 1

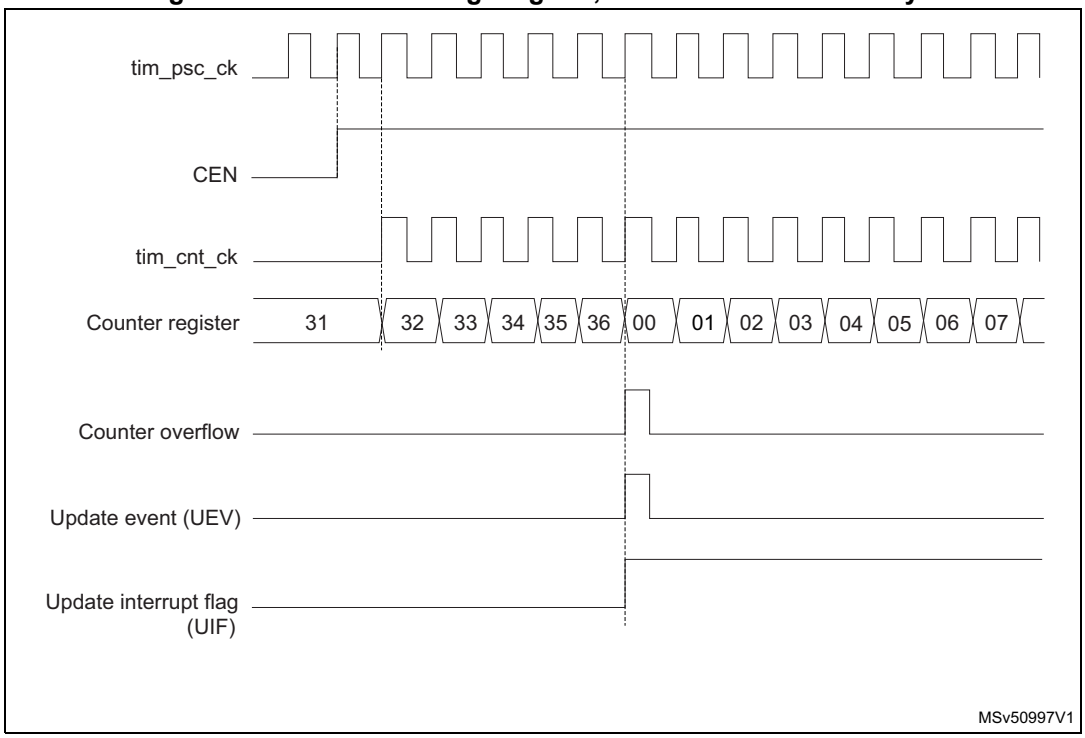


Figure 488. Counter timing diagram, internal clock divided by 2

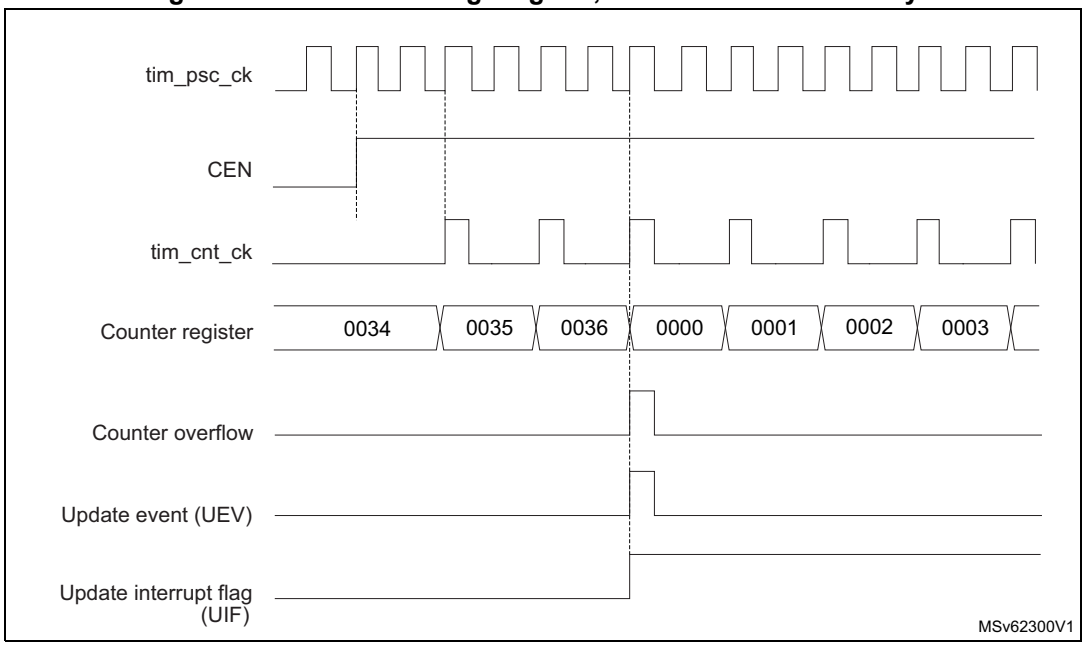


Figure 489. Counter timing diagram, internal clock divided by 4

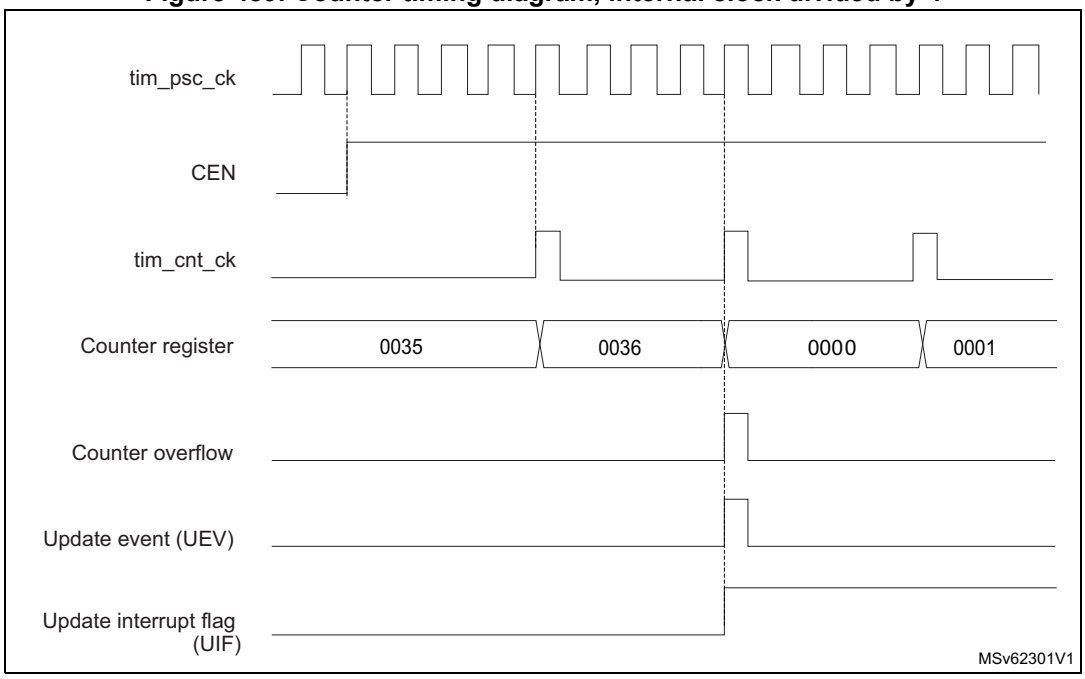


Figure 490. Counter timing diagram, internal clock divided by N

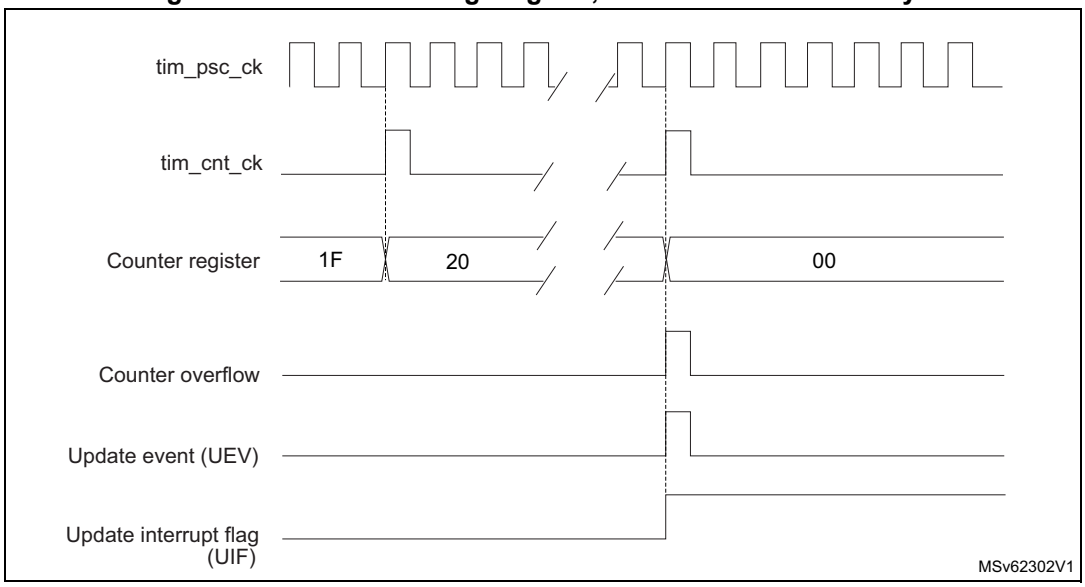


Figure 491. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

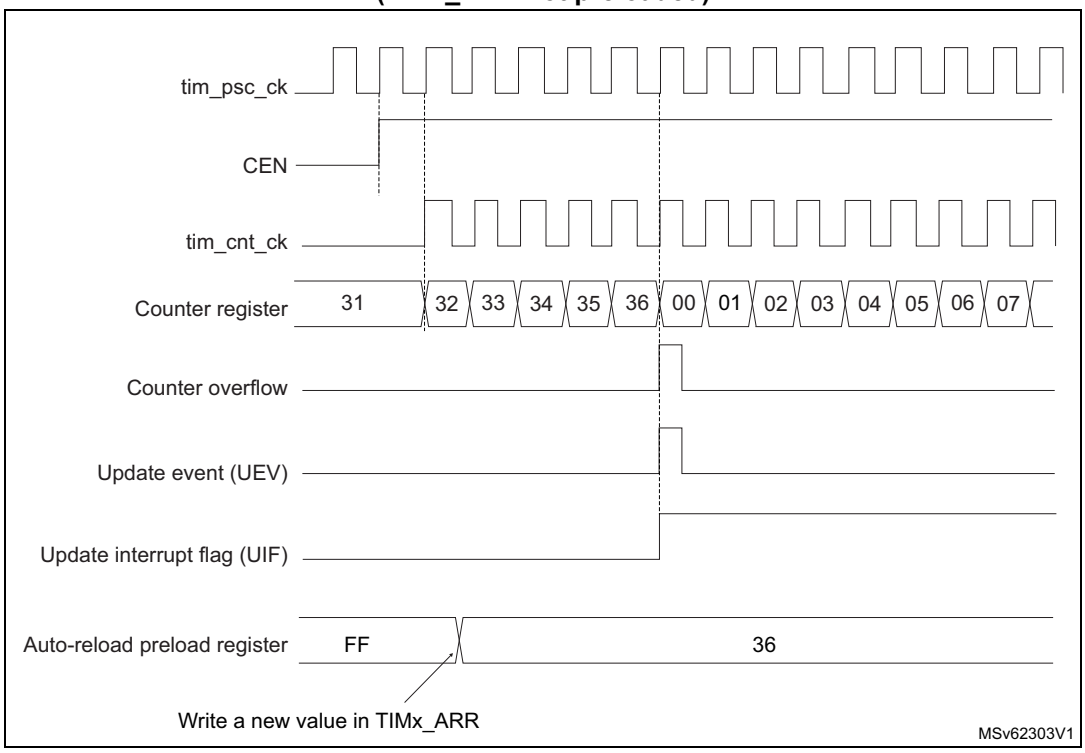
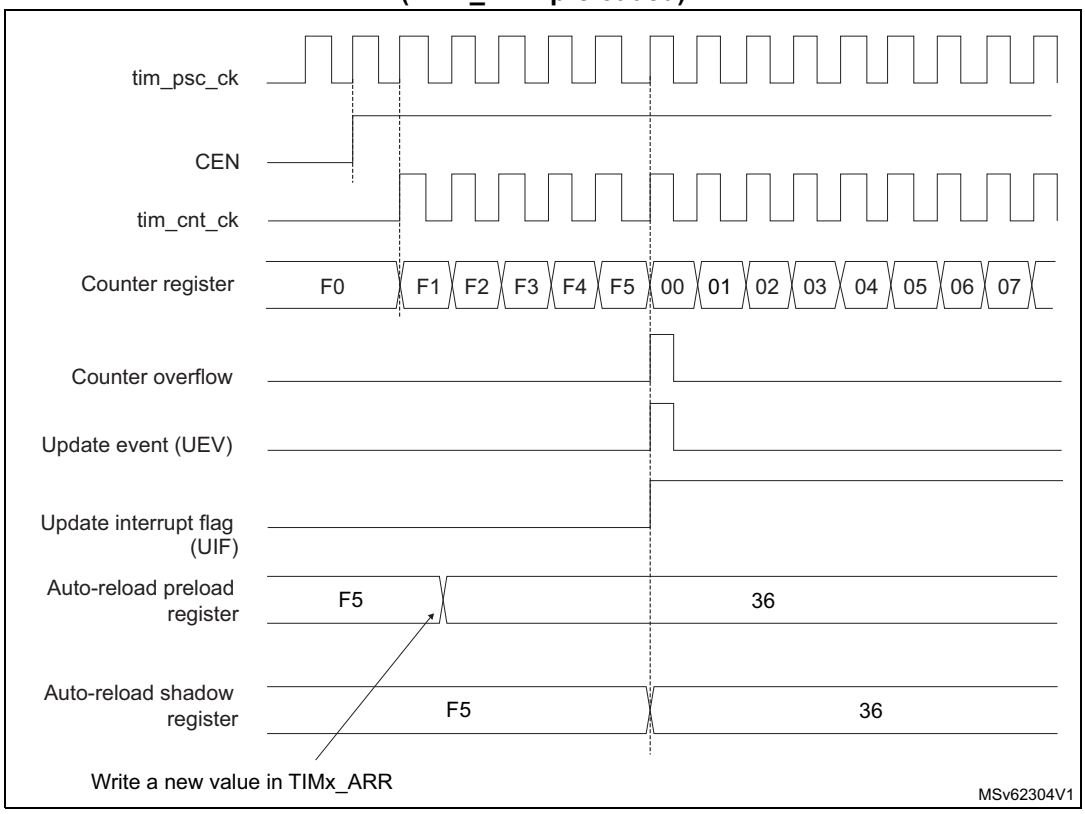


Figure 492. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



38.4.5 Repetition counter

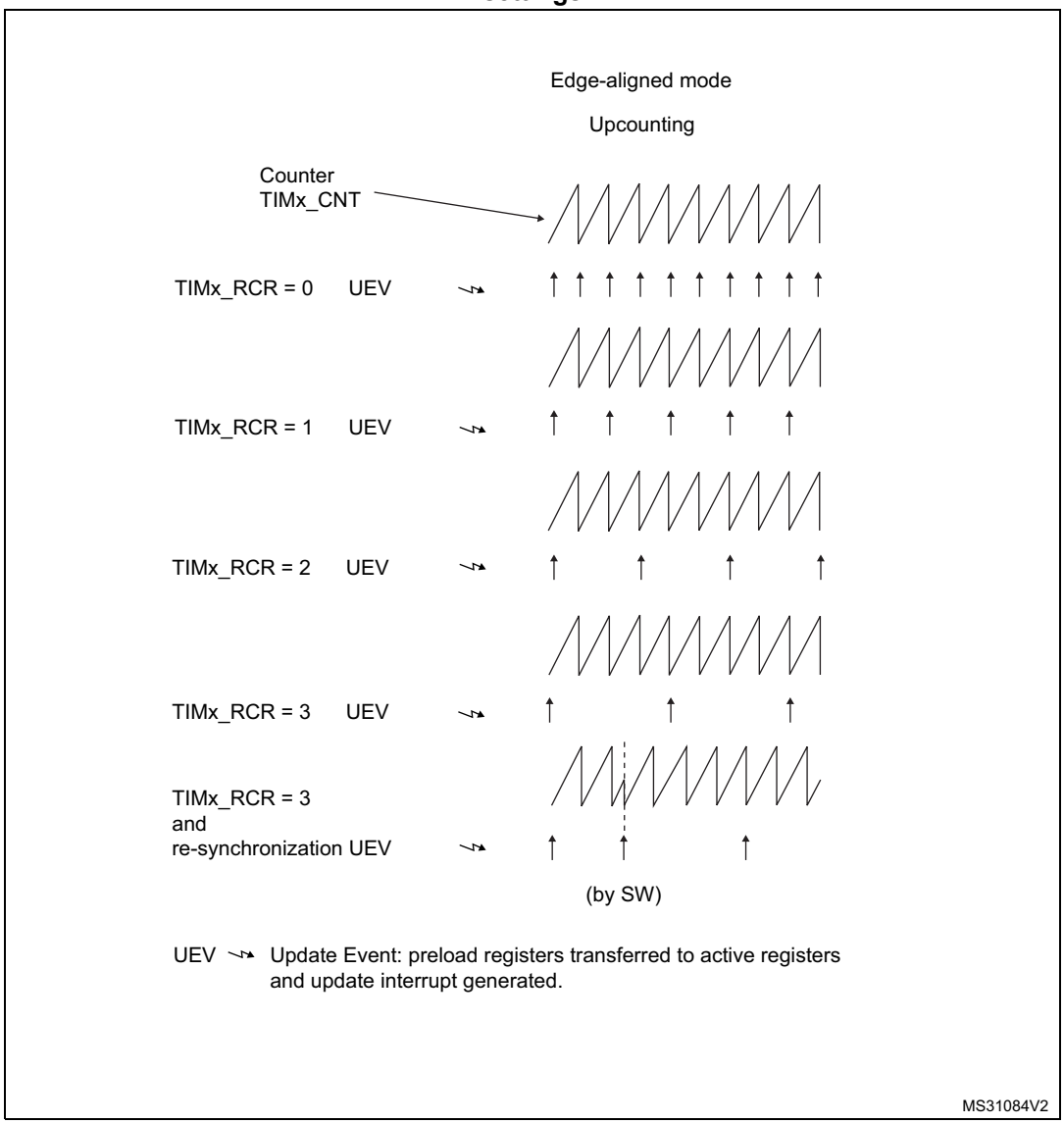
Section 38.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 493*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 493. Update rate examples depending on mode and TIMx_RCR register settings



38.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer* for more details.

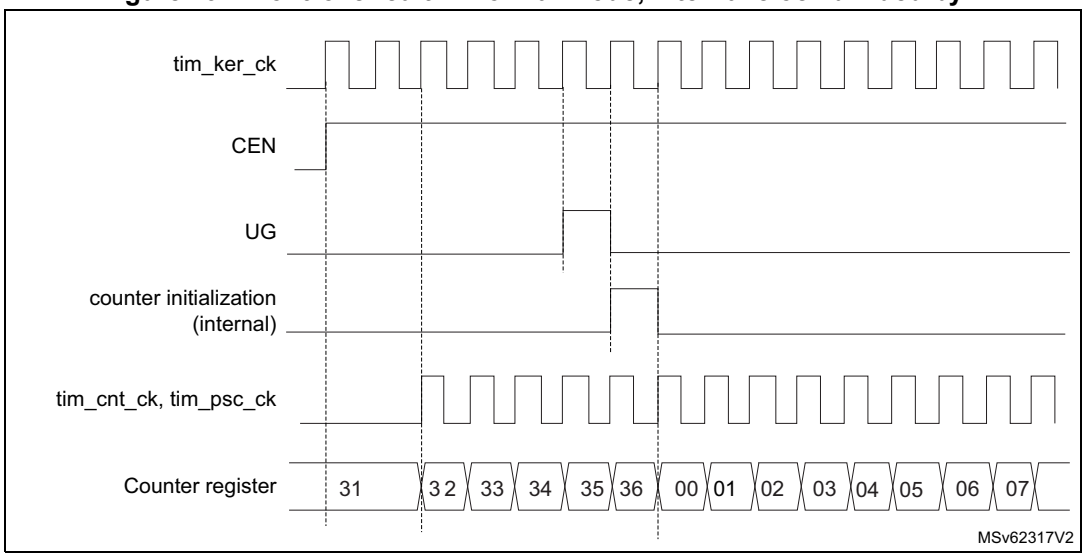
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 494 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

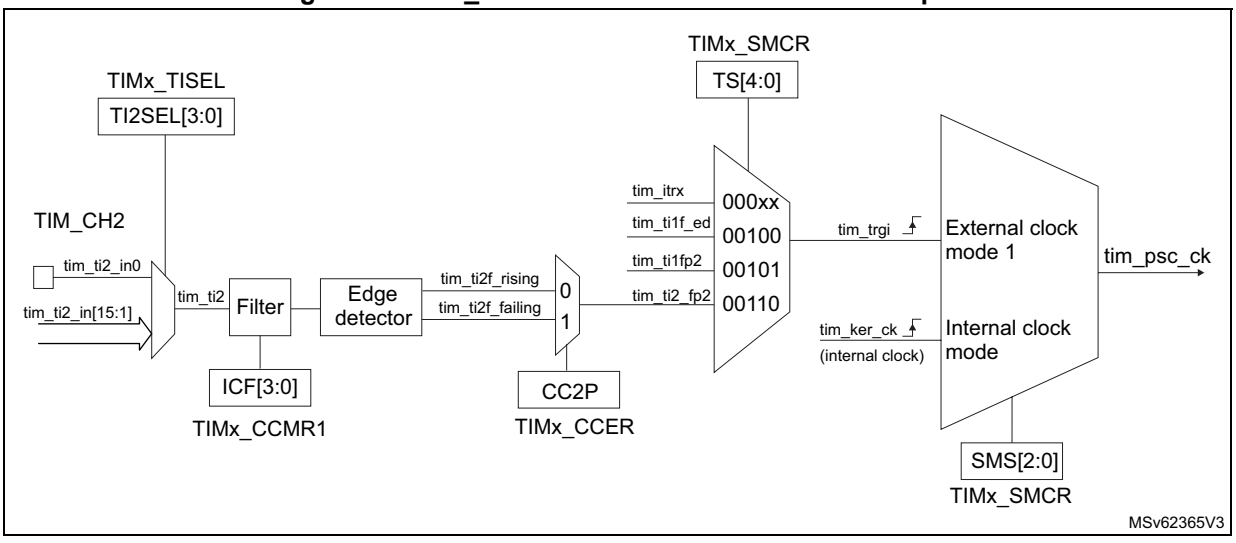
Figure 494. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 495. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

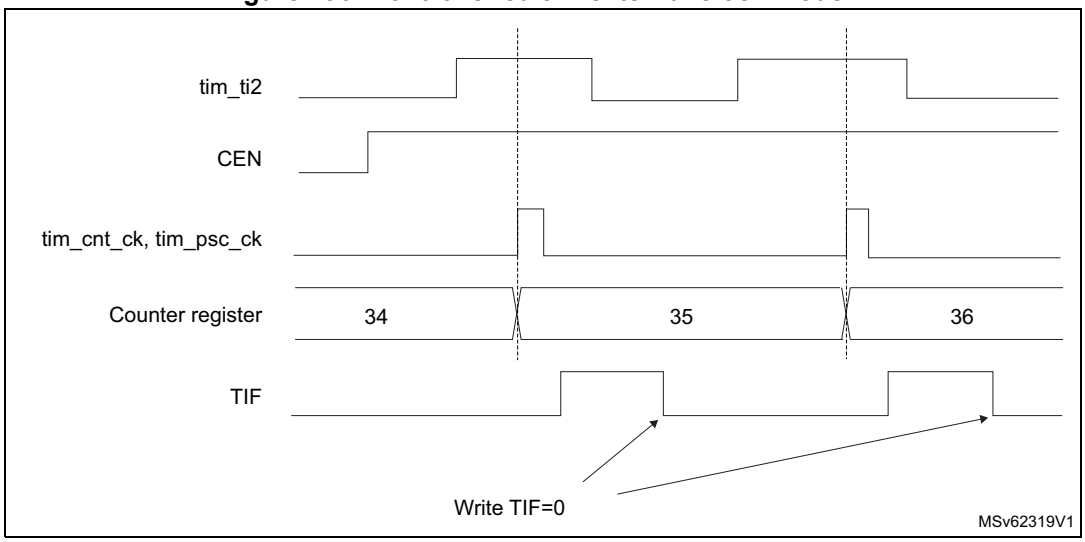
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 496. Control circuit in external clock mode 1



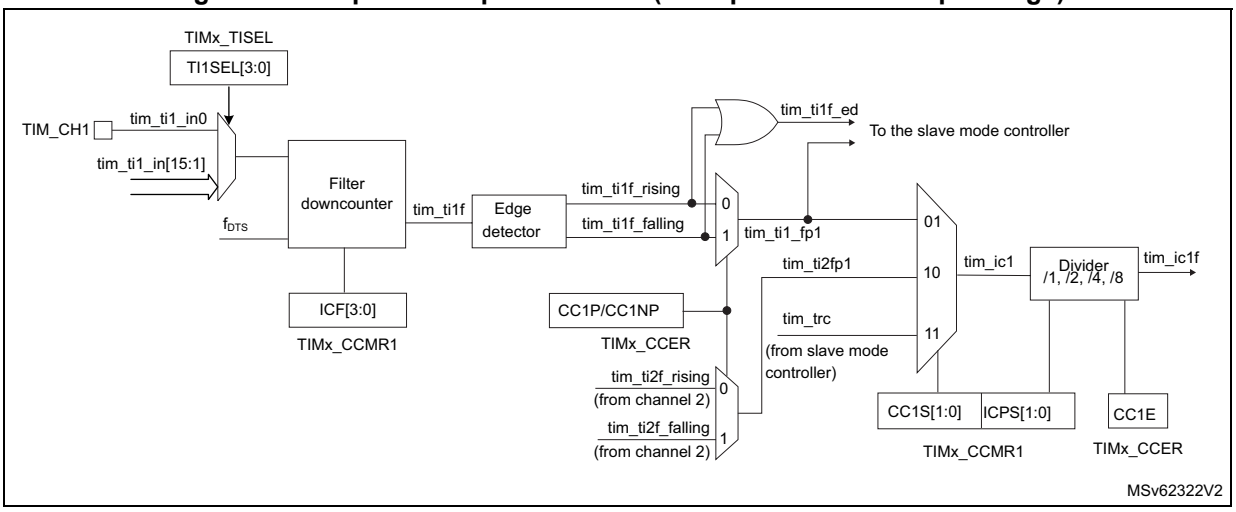
38.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 497 to *Figure 500* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 497. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 498. Capture/compare channel 1 main circuit

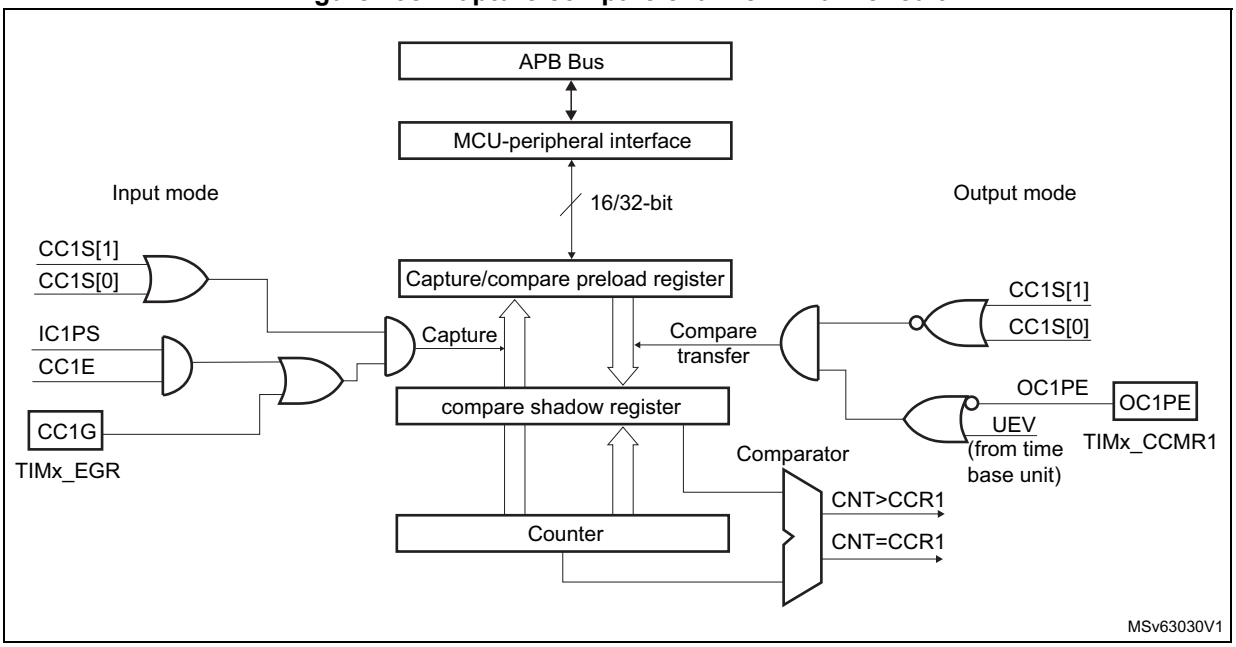


Figure 499. Output stage of capture/compare channel (channel 1)

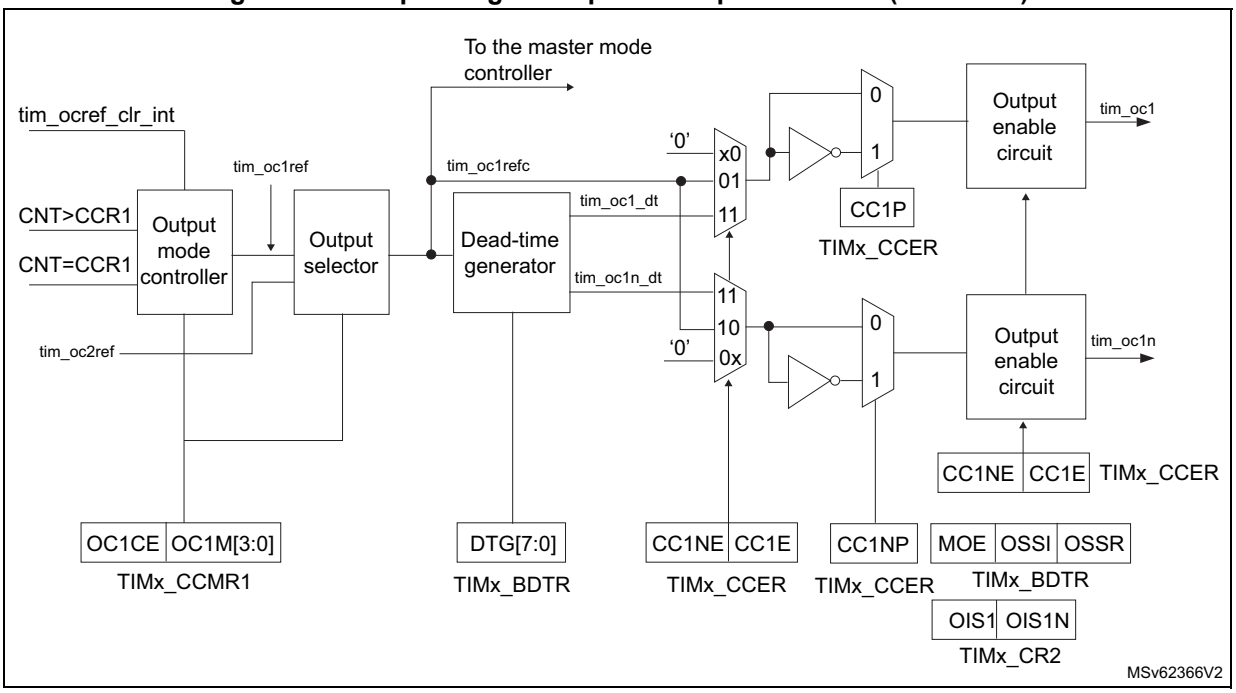
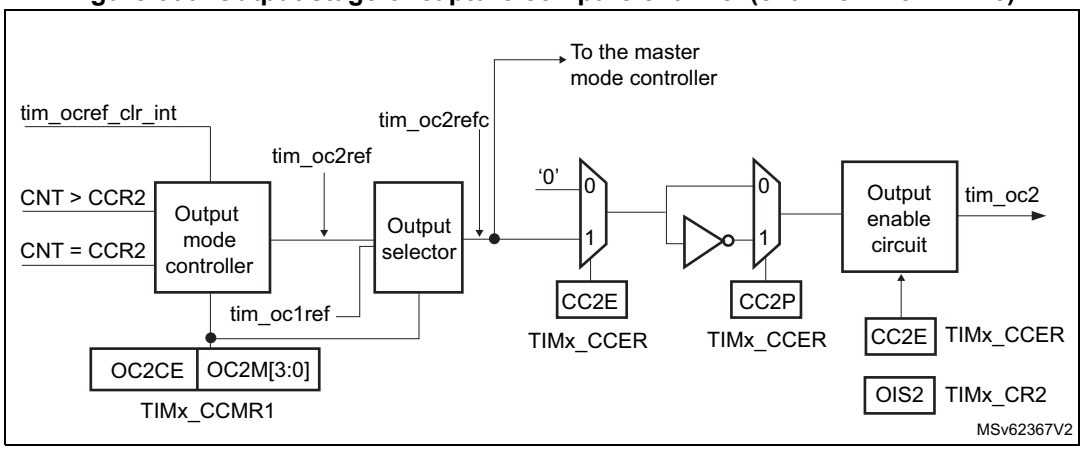


Figure 500. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

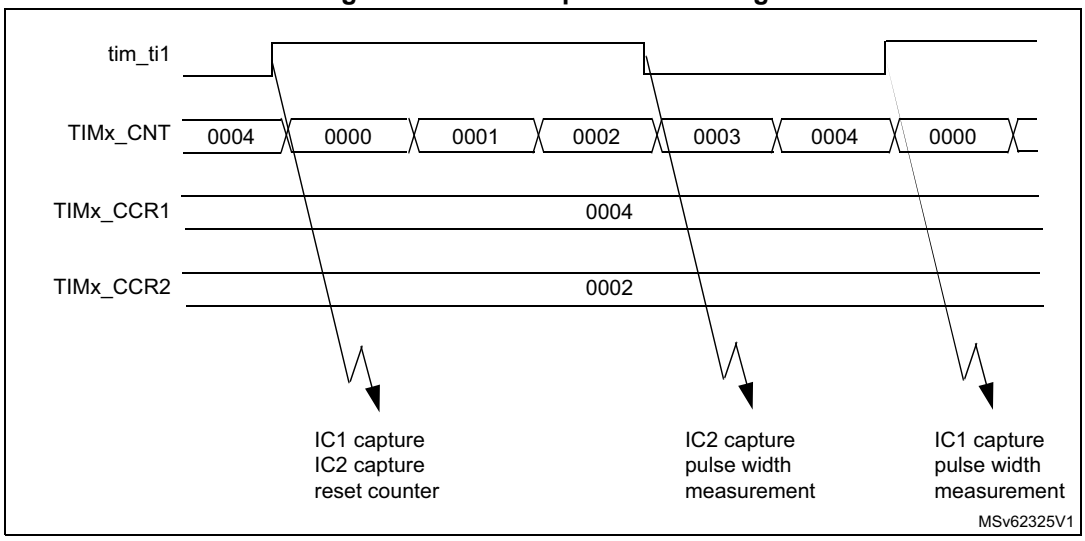
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 501. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

38.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

38.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

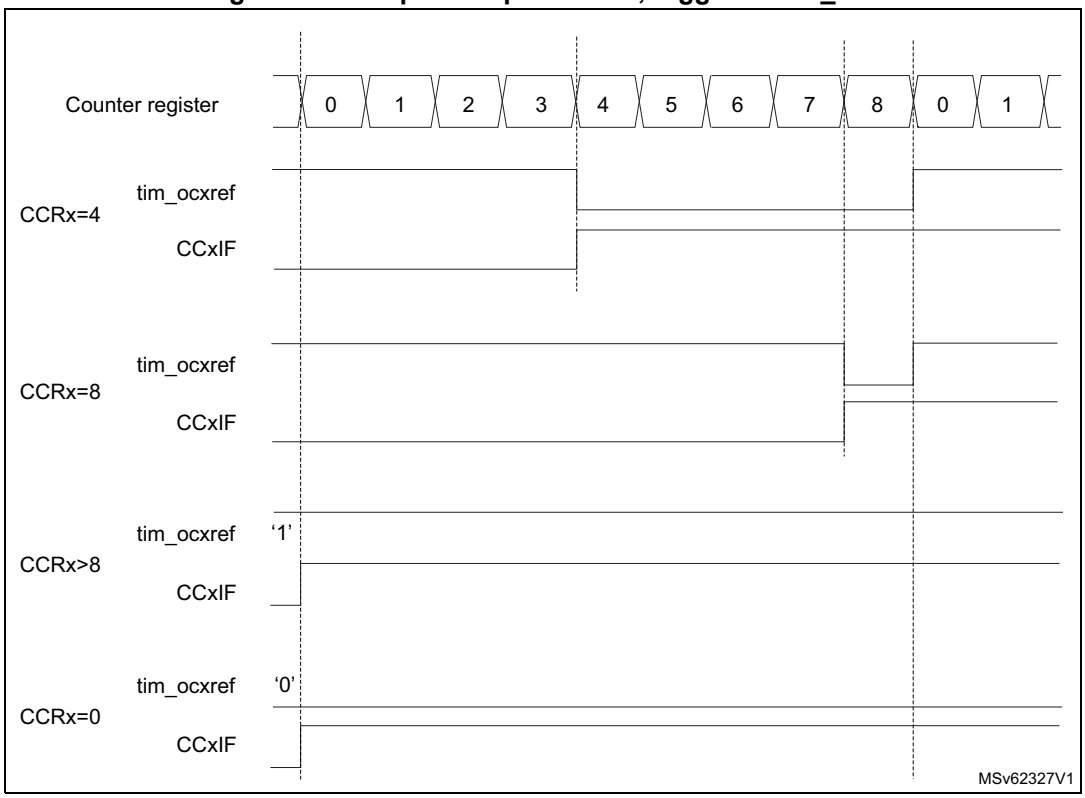
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 502*.

Figure 502. Output compare mode, toggle on tim_oc1



38.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

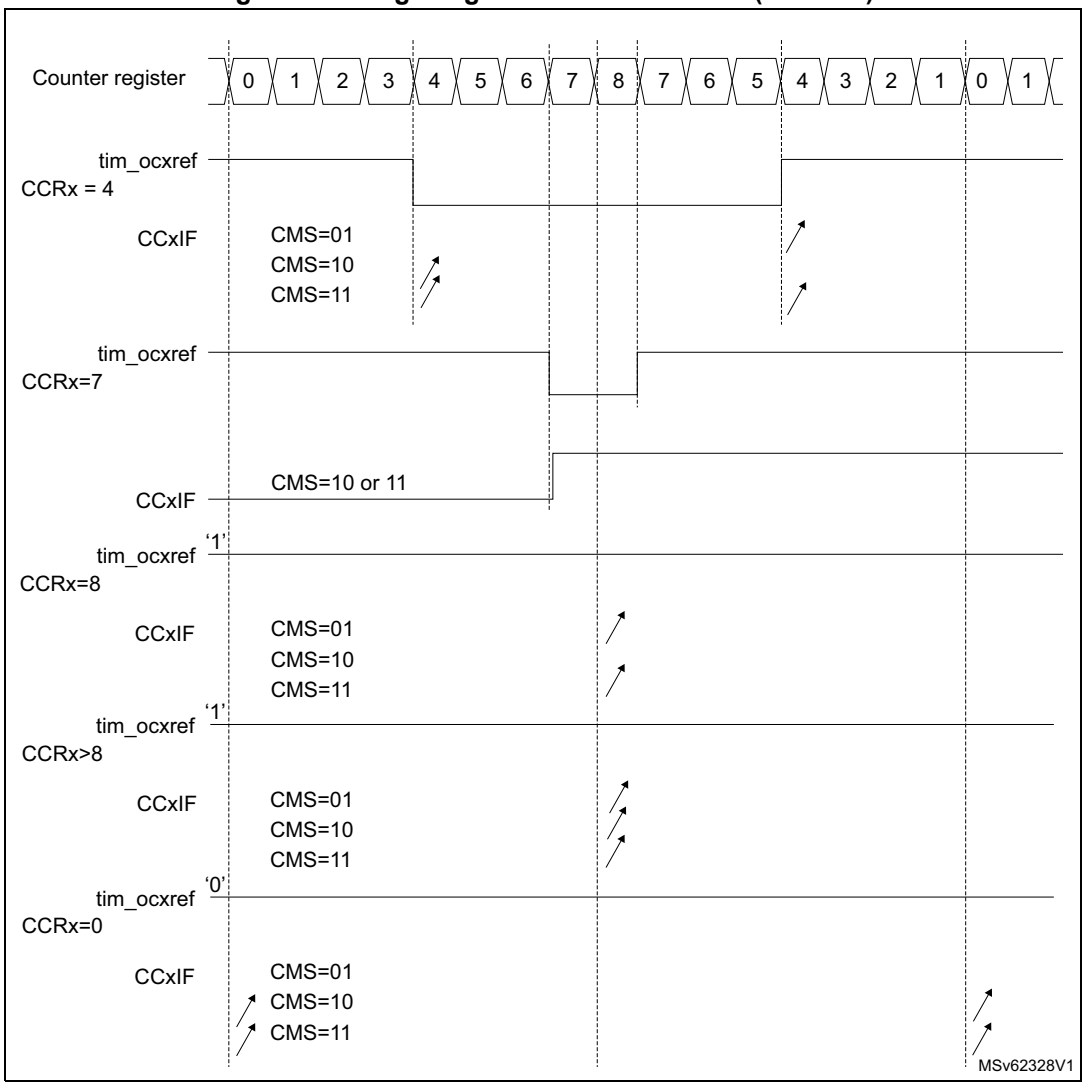
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in

TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 503* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 503. Edge-aligned PWM waveforms (ARR = 8)

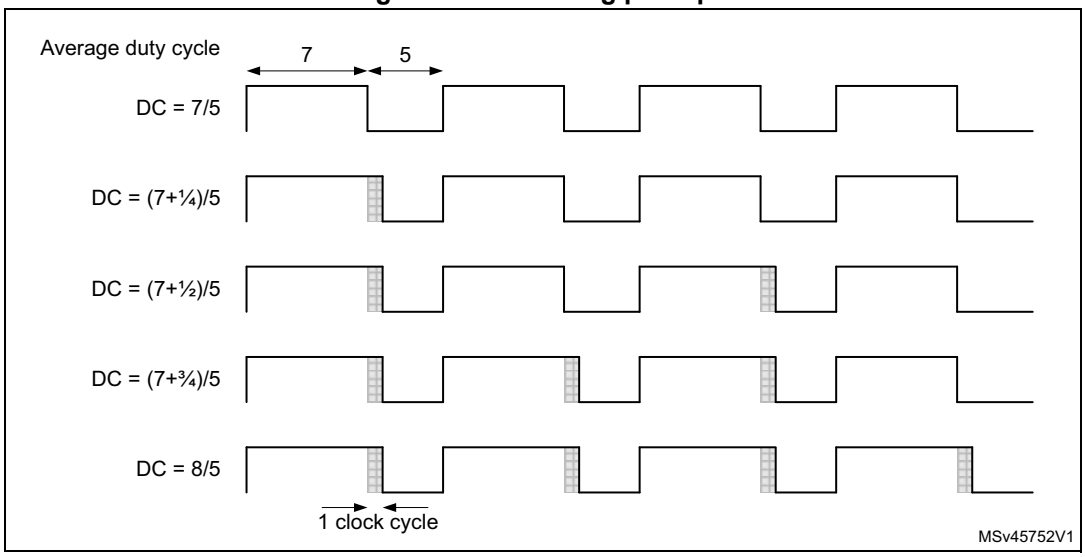


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 504* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 504. Dithering principle

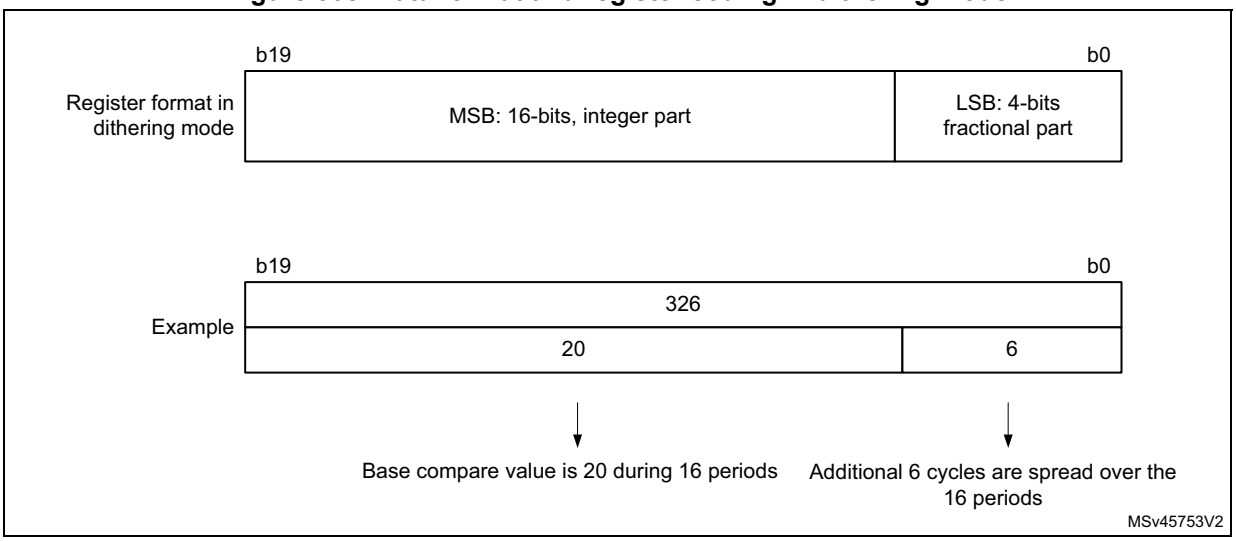


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 505* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 505. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

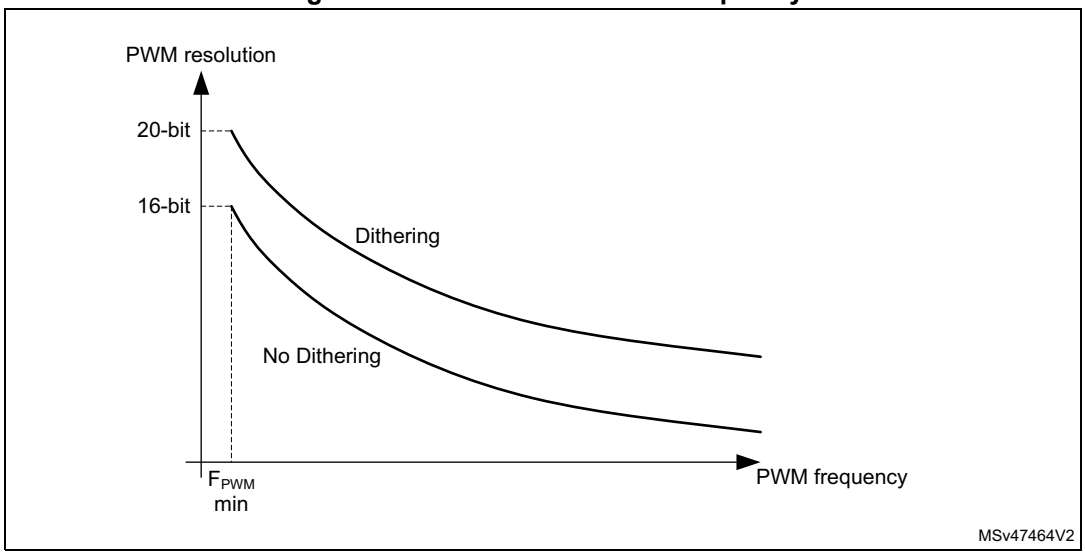
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

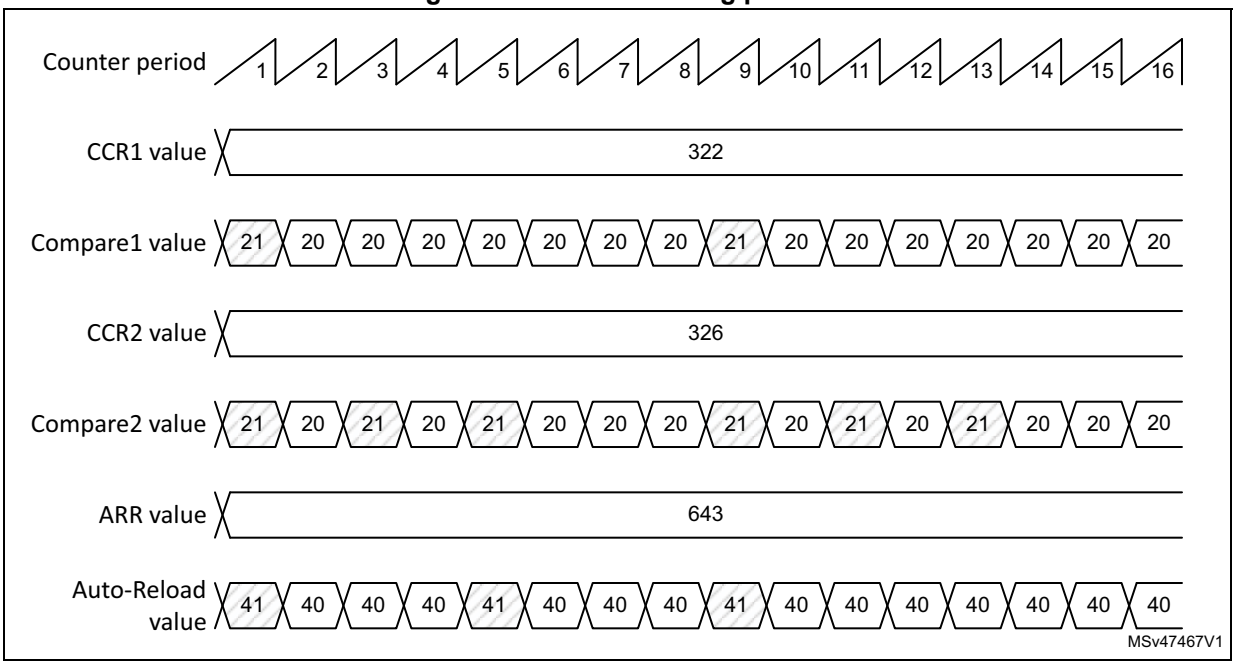
As shown on the *Figure 506* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 506. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 507* below.

Figure 507. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 397* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 397. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 397. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

38.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

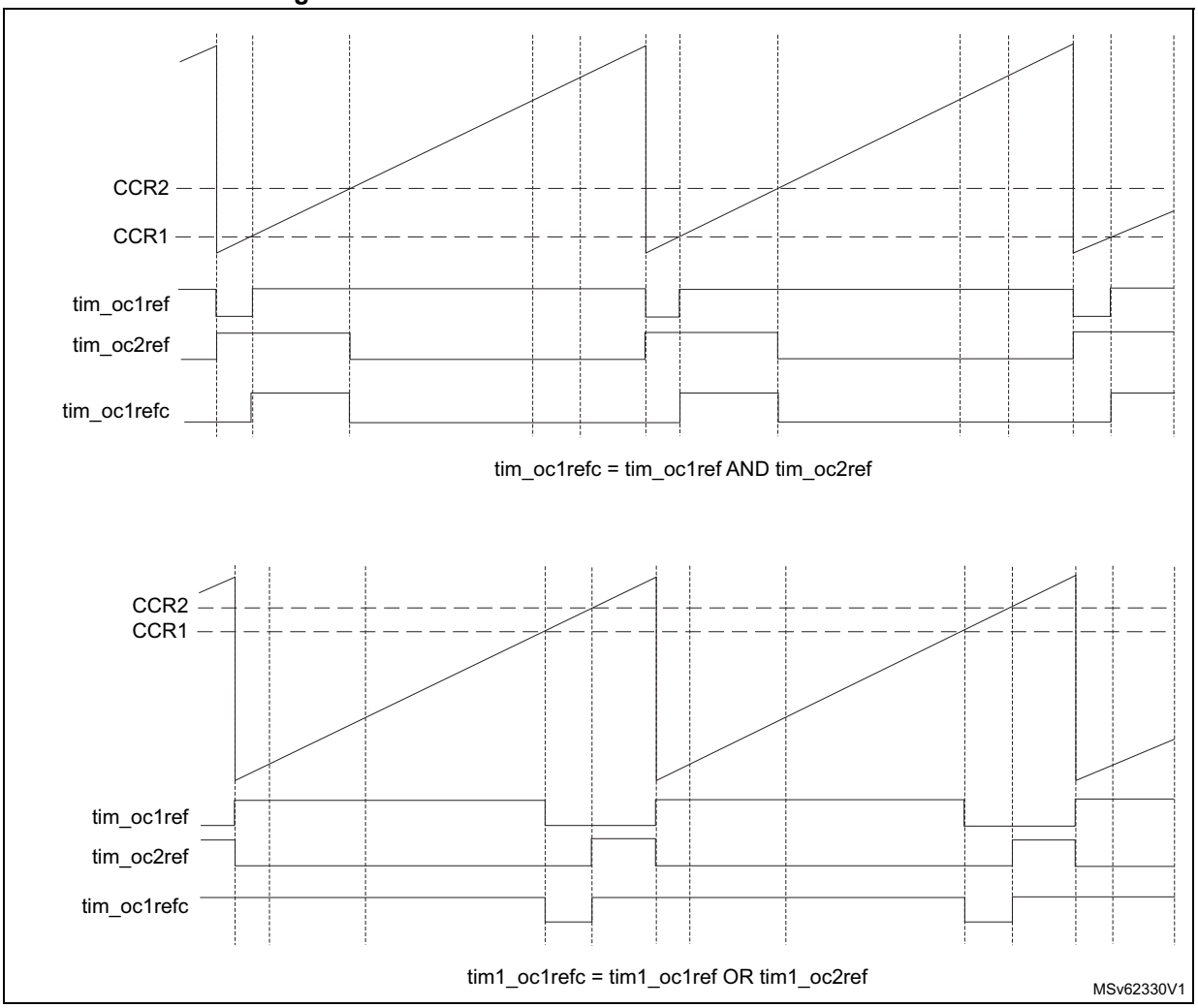
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Table 508 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 508. Combined PWM mode on channel 1 and 2



38.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 404: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

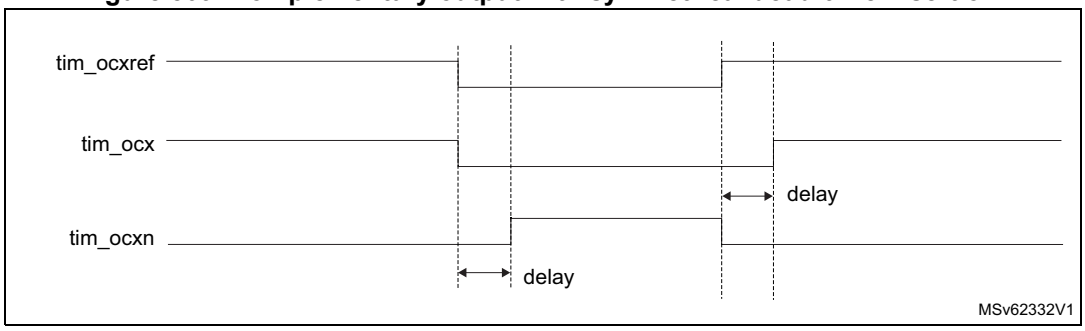
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 509. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 510*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 510. Asymmetrical deadtime

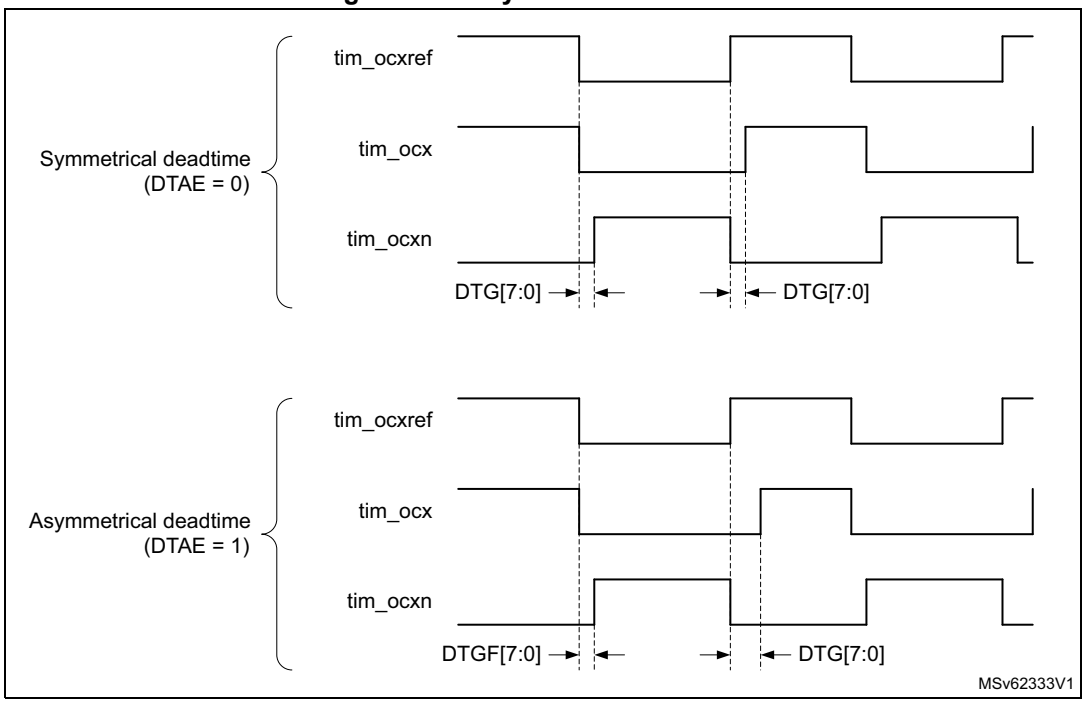


Figure 511. Dead-time waveforms with delay greater than the negative pulse.

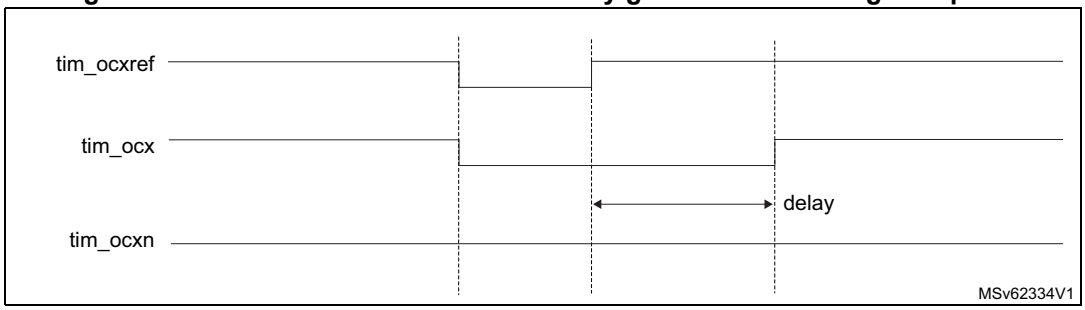
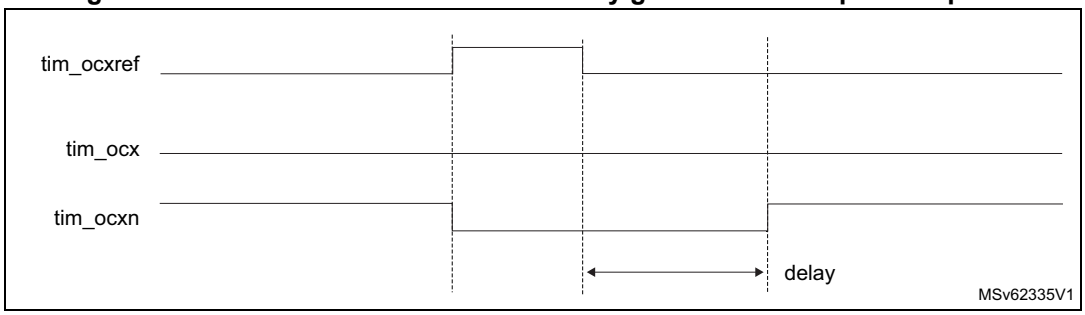


Figure 512. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 38.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

38.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 404: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

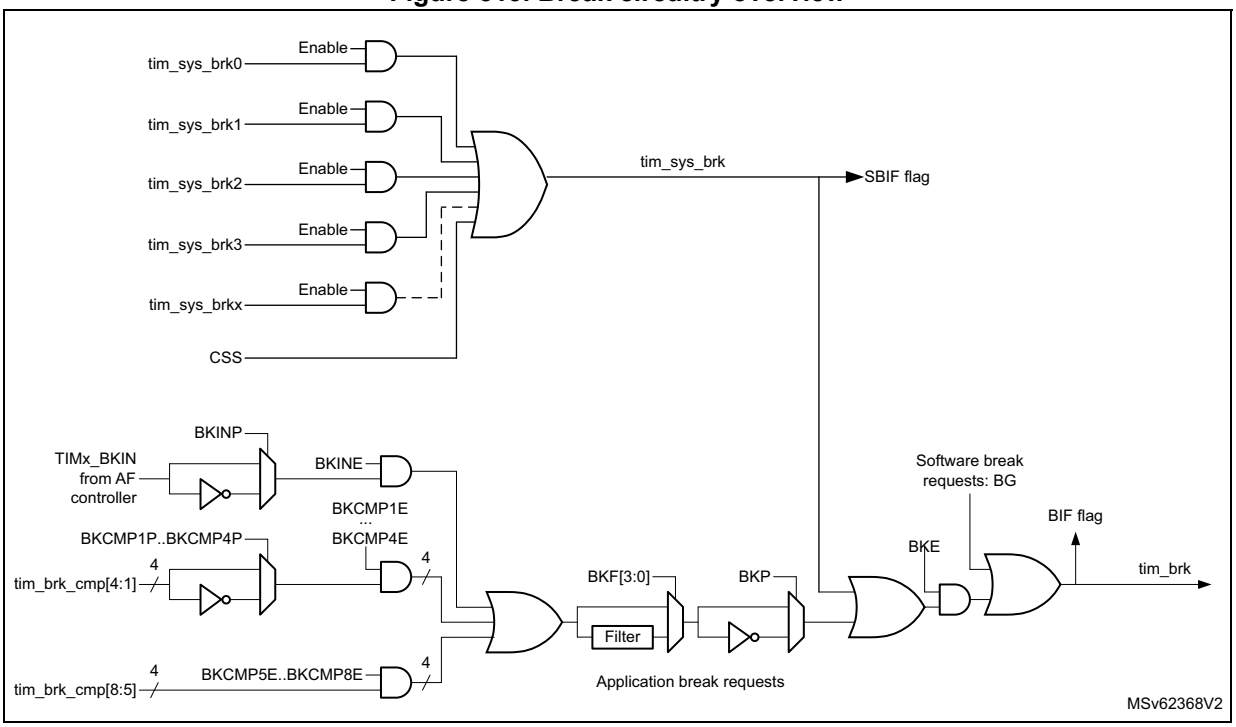
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 513* below.

Figure 513. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

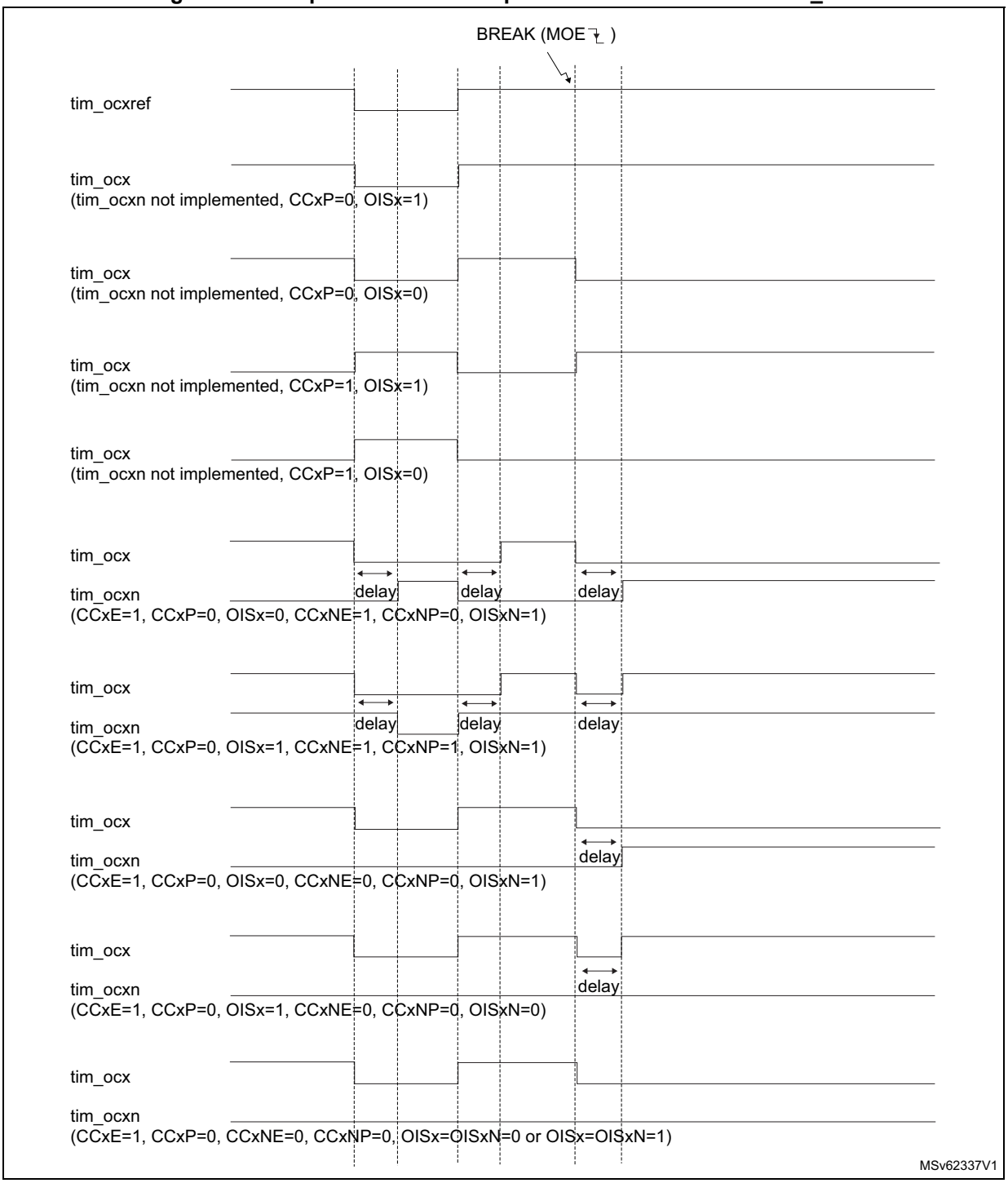
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 38.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 514* shows an example of behavior for the outputs in response to a break.

Figure 514. Output behavior in response to a break event on tim_brk



38.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 515*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 398*).

Table 398. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

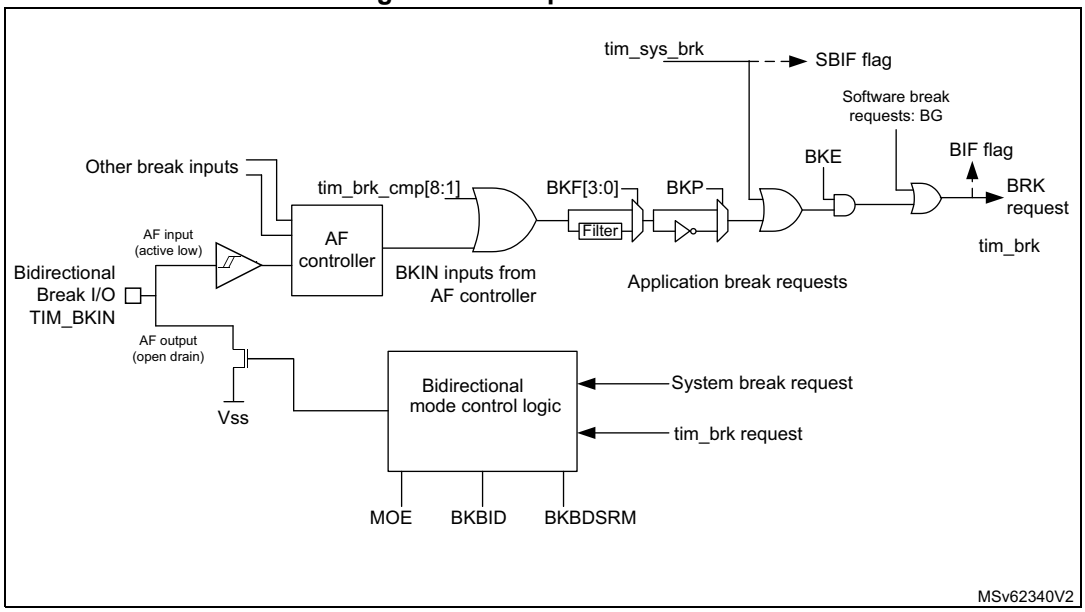
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 515. Output redirection

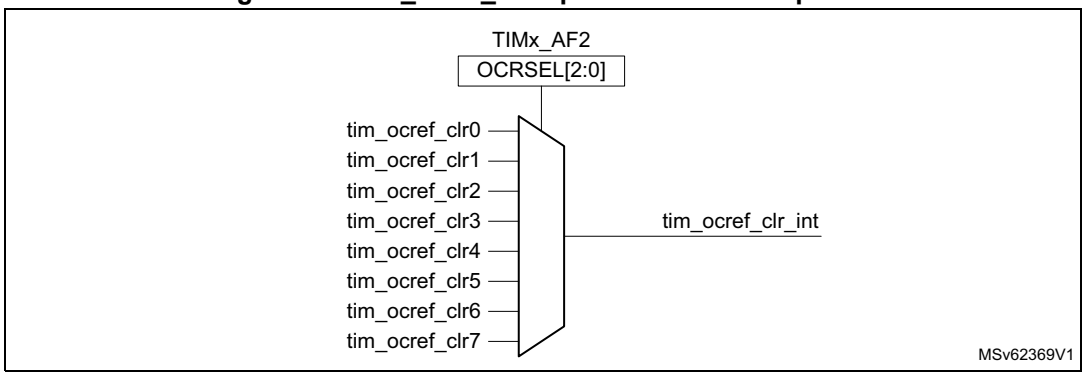


38.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 509* below.

Figure 516. tim_ocref_clr input selection multiplexer



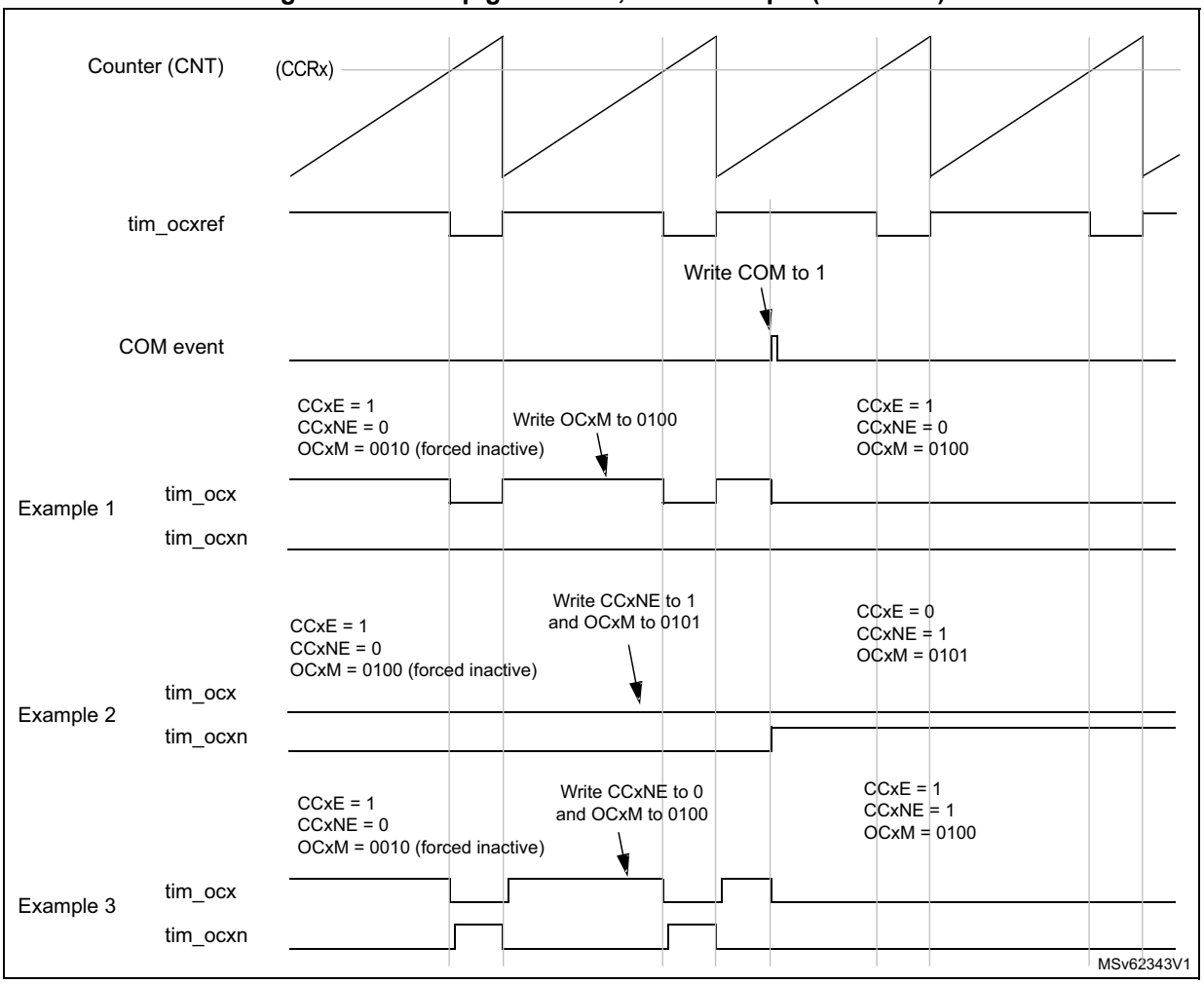
38.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 517* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 517. 6-step generation, COM example (OSSR = 1)



38.4.19 One-pulse mode

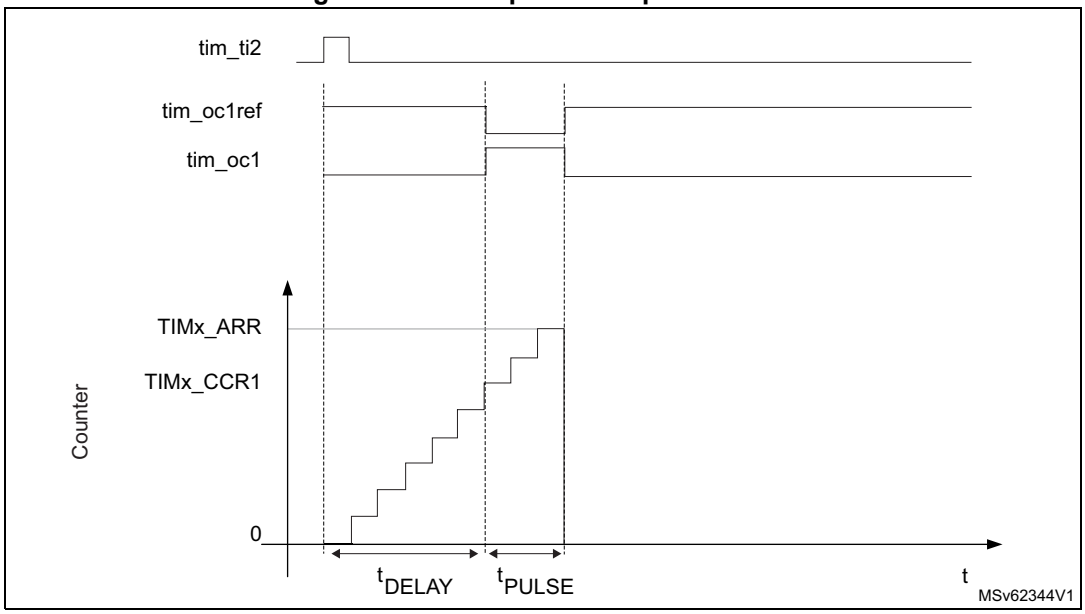
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

Figure 518. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.4.20 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 38.4.19*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

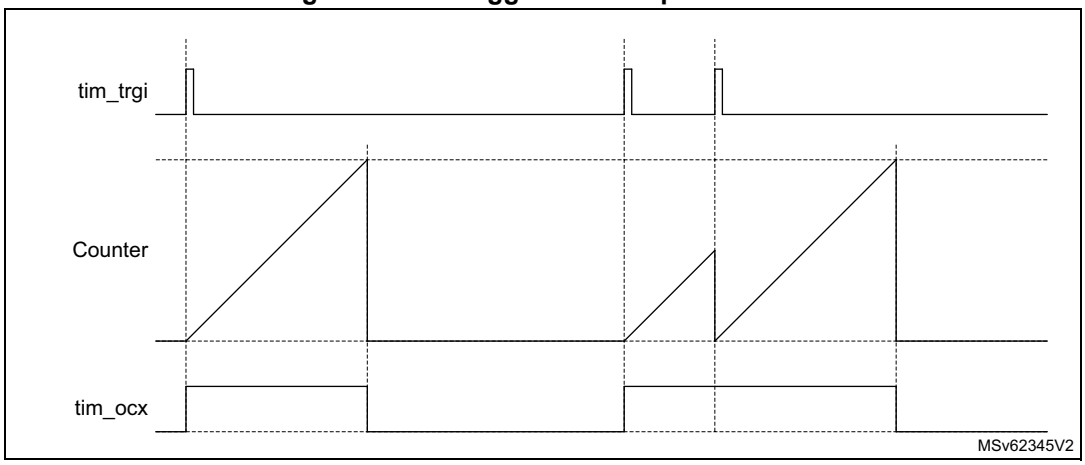
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 519. Retriggerable one pulse mode



38.4.21 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

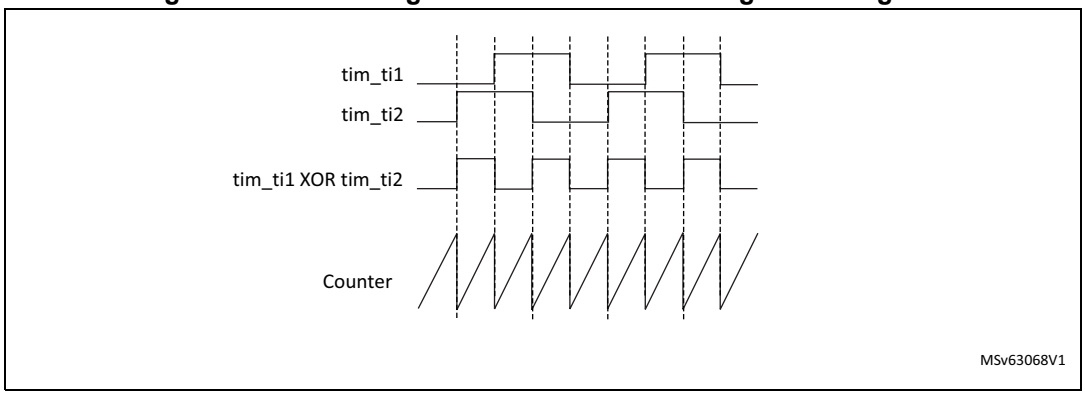
There is no latency between the assertions of the UIF and UIFCPY flags.

38.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 520*.

Figure 520. Measuring time interval between edges on 2 signals



38.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

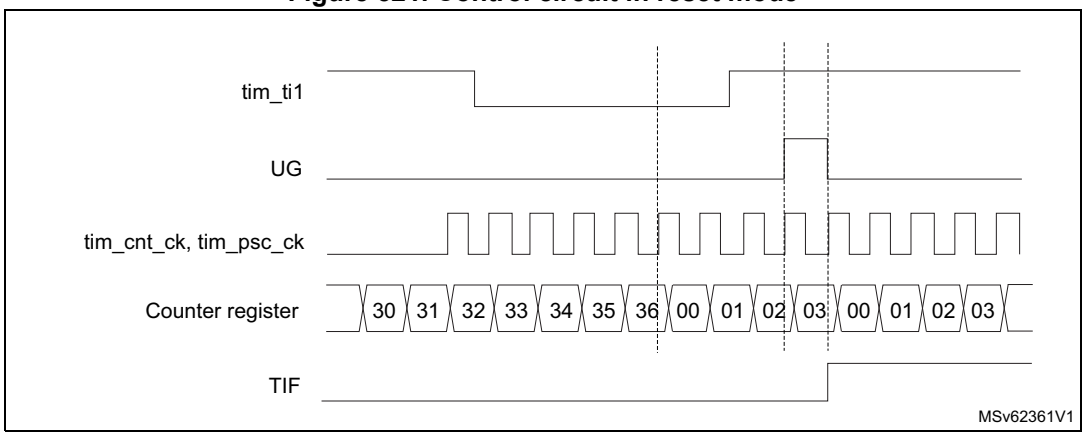
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 521. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

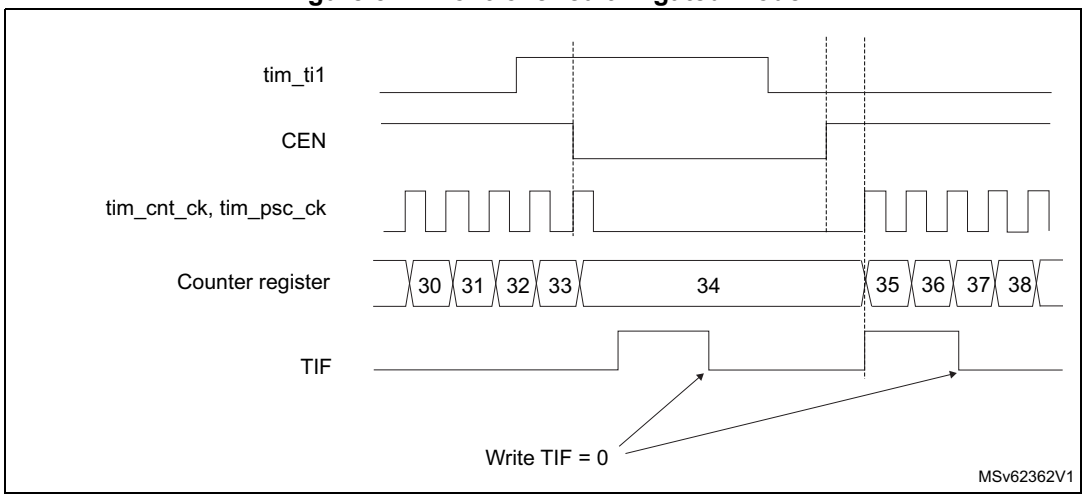
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 522. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

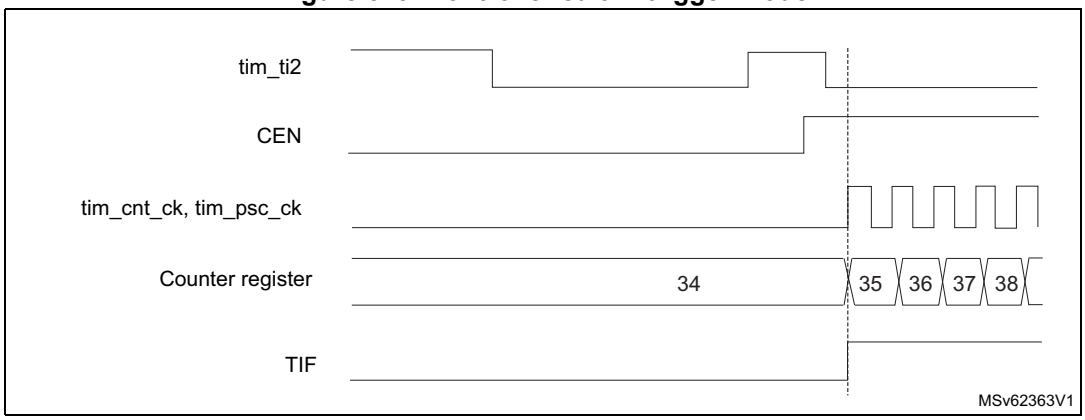
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 523. Control circuit in trigger mode



Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

38.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

38.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

38.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 36.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

38.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.4.29 ADC synchronization (TIM15 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semi-synchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the 2 requirements below is met (depending on the selected trigger source):

1. the counter period must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. the compare value must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

If none of the 2 above requirement are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

38.4.30 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 38.8.20: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.4.31 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 399*.

Table 399. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_com_dma(1)	Commutation (COM)	COMDE
tim_trgi_dma(1)	Trigger	TDE

1. Available for TIM15 only.

38.4.32 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

38.5 TIM15/TIM16/TIM17 low-power modes

Table 400. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

38.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 401*.

Table 401. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

38.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 38.4.29* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2 = 0 when MOE = 0

1: tim_oc2 = 1 when MOE = 0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

38.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
							rw			rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded.

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw				rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM15_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 38.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

38.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

38.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT = TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

38.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 402* for details.

Table 402. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
tim_ocx = 0
tim_ocxn = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>tim_ocxref + Polarity + dead-time</td><td>Complementary to tim_ocxref (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="6">0</td><td>0</td><td rowspan="6">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="5">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP
Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

38.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

38.7.12 TIM15 autoreload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 38.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

38.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

38.7.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1586*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1586*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 38.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1586*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS} = 125 ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

38.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

38.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

38.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

38.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

0110: COM

0111: Trigger

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

38.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

38.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 403. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM15_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIM15_CR2**	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value				0																		0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0				0								0	0	0	0			0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0				0	0	0	0	0			0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0

Table 403. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	**TIM15_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM15_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM15_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM15_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM15_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															

Table 403. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	**TIM15_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIM15_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM15_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM15_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM15_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM15_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

38.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

38.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

38.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

38.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

38.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

38.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 404* for details.

Table 404. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CC1E bit</th><th>CC1NE bit</th><th>tim_oc1 output state</th><th>tim_oc1n output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0
tim_oc1n = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1n = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>tim_oc1ref + Polarity + dead-time</td><td>Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_oc1 = CC1P</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Off-State (output enabled with inactive state)
tim_oc1n = CC1NP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP
Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

38.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

38.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 38.4.3: Time-base unit on page 1528* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

38.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 38.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

38.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

38.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 38.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

38.8.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32 EN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HSE32EN**: HSE divided by 32 enable

This bit enables the HSE divider by 32 for the tim_ti1_in3. See *Table 391: Interconnect to the tim_ti1 input multiplexer* for details.

0: HSE divided by 32 disabled

1: HSE divided by 32 enabled

Bit 0 Reserved, must be kept at reset value.

38.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

38.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

38.8.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 405. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 405. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68	TIMx_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32EN	Res.
	Reset value																															0	
0x6C - 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 Low-power timer (LPTIM)

39.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

39.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

39.3 LPTIM implementation

The table below describes LPTIM implementation on STM32U3 series devices. The full set of features is implemented in LPTIM1, LPTIM2 and LPTIM3. LPTIM4 supports a smaller set of features.

Table 406. STM32U3 series LPTIM features

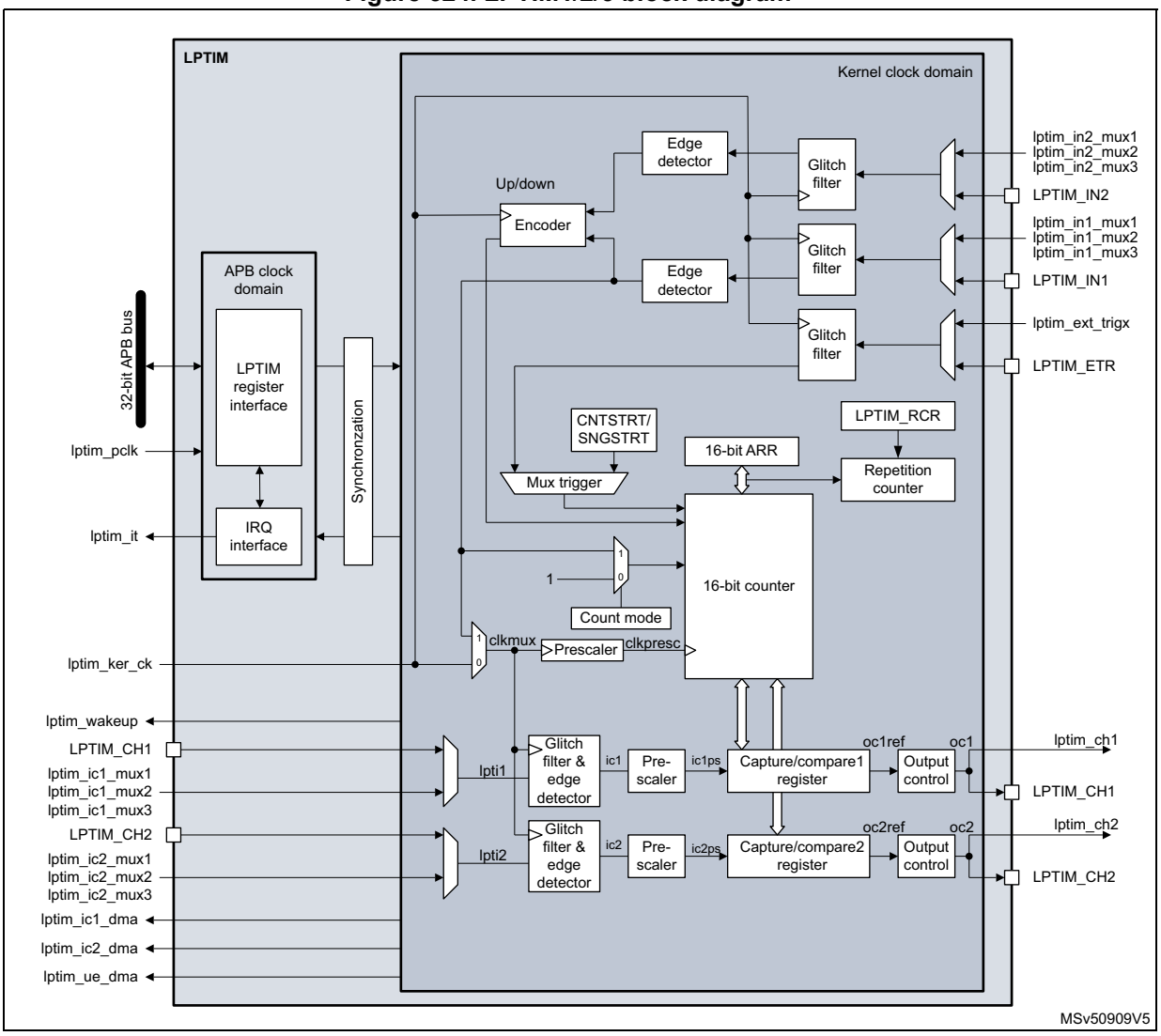
LPTIM modes/features[1]	LPTIM1	LPTIM2	LPTIM3	LPTIM4
Encoder mode	X	X	-	-
PWM mode	X	X	X	X
Input Capture	X	X	X	-
Number of channels	2	2	2	-
Number of DMA requests	3	3	3	-
Wake-up from Stop mode	X[2]	X[3]	X[2]	X[2]
Autonomous mode	X	X	X	-

1. X = supported.
2. Wake-up supported from Stop 0, Stop 1, and Stop 2 modes.
3. Wake-up supported from Stop 0 and Stop 1 modes.

39.4 LPTIM functional description

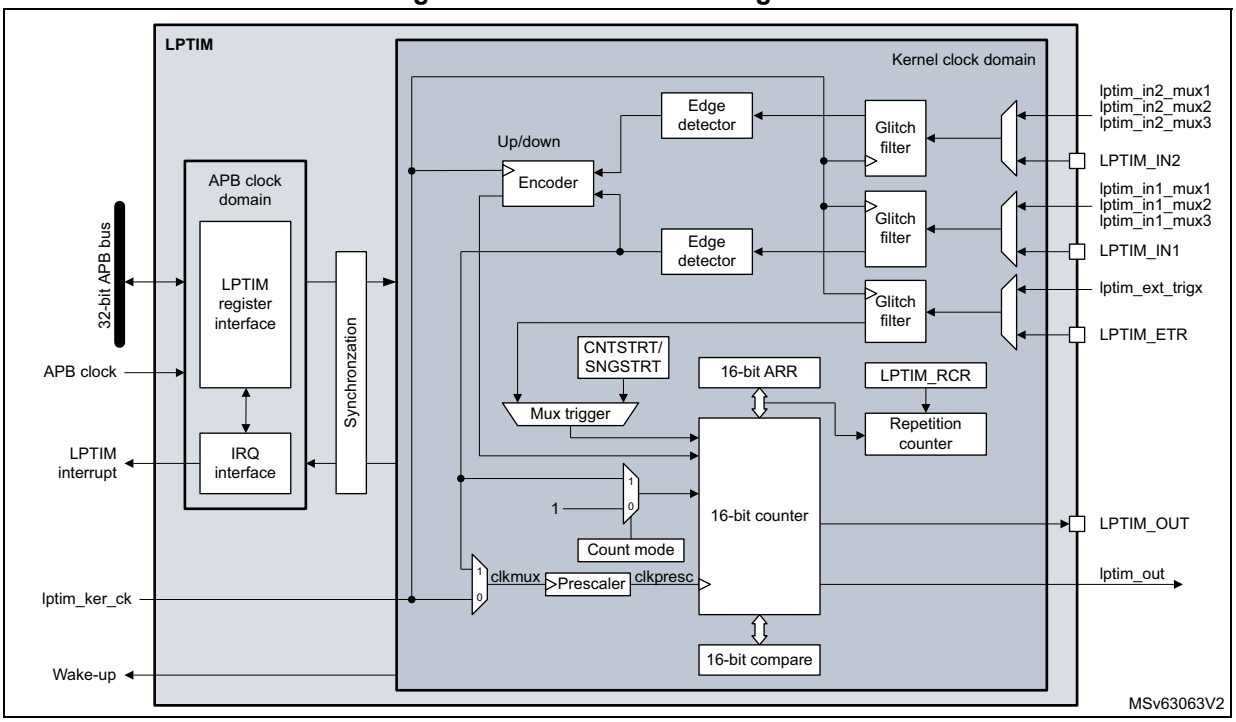
39.4.1 LPTIM block diagram

Figure 524. LPTIM1/2/3 block diagram(1)



1. Some I/Os may not be available, refer to *Section 39.4.2: LPTIM pins and internal signals*.

Figure 525. LPTIM4 block diagram(1)



1. Some I/Os may not be available, refer to *Section 39.4.2: LPTIM pins and internal signals*.

39.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 407. LPTIM1/2/3 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM input 1 from GPIO pin on mux input 0
LPTIM_IN2(1)	Digital input	LPTIM input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIOpin

1. LPTIM3 has only the input 1 (no input 2).

Table 408. LPTIM4 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM output GPIO pin

Table 409. LPTIM1/2/3 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1(1)	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2(1)	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3(1)	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

1. LPTIM3 has only the input 1(no input 2).

Table 410. LPTIM4 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

39.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 411. LPTIM1/2/3/4 external trigger connections

TRIGSEL	External trigger			
	LPTIM1	LPTIM2	LPTIM3	LPTIM4
lptim_ext_trig0	GPIO			
lptim_ext_trig1	rtc_alra_trg			
lptim_ext_trig2	rtc_alrb_trg			
lptim_ext_trig3	tamp_trg1			gpdma_ch3_tcf
lptim_ext_trig4	tamp_trg2	gpdma_ch0_tcf	gpdma_ch1_tcf	tamp_trg2
lptim_ext_trig5	gpdma_ch2_tcf	gpdma_ch4_tcf	tamp_trg3	tamp_trg3
lptim_ext_trig6	comp1_out			
lptim_ext_trig7	comp2_out			

Table 412. LPTIM1/2/3/4 input 1 connections

lptim_in1_mux	LPTIM1/2/3/4 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	comp1_out
lptim_in1_mux2	Reserved
lptim_in1_mux3	Reserved

Table 413. LPTIM1/2 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	comp2_out
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 414. LPTIM1 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	comp1_out
lptim_ic1_mux2	comp2_out
lptim_ic1_mux3	i3c2_ibiack

Table 415. LPTIM2/3 input capture 1 connections

lptim_ic1_mux	LPTIM2/3 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	comp1_out
lptim_ic1_mux2	comp2_out
lptim_ic1_mux3	i3c1_ibiack

Table 416. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	Reserved

Table 417. LPTIM2 input capture 2 connections

lptim_ic2_mux	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	HSI/256
lptim_ic2_mux2	MSIS/1024
lptim_ic2_mux3	MSIS/4

Table 418. LPTIM3 input capture 2 connections

lptim_ic2_mux	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	i3c2_ibiack
lptim_ic2_mux2	Reserved
lptim_ic2_mux3	Reserved

39.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout

function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

39.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

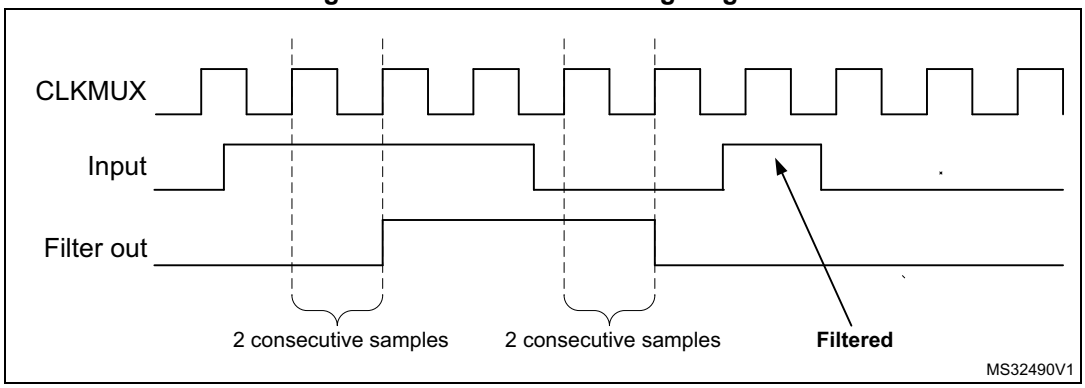
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 526* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 526. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

39.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 419. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

39.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

39.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

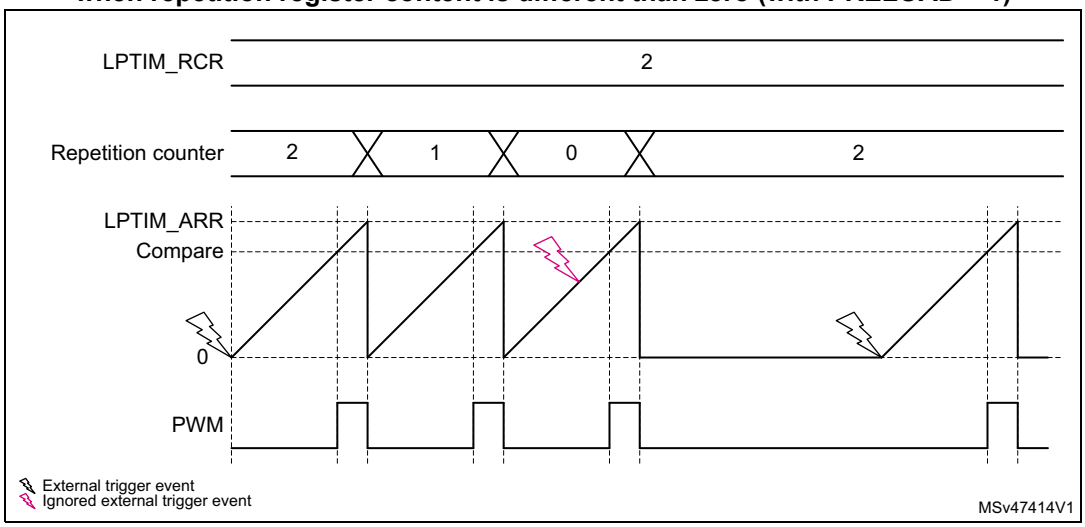
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 527*.

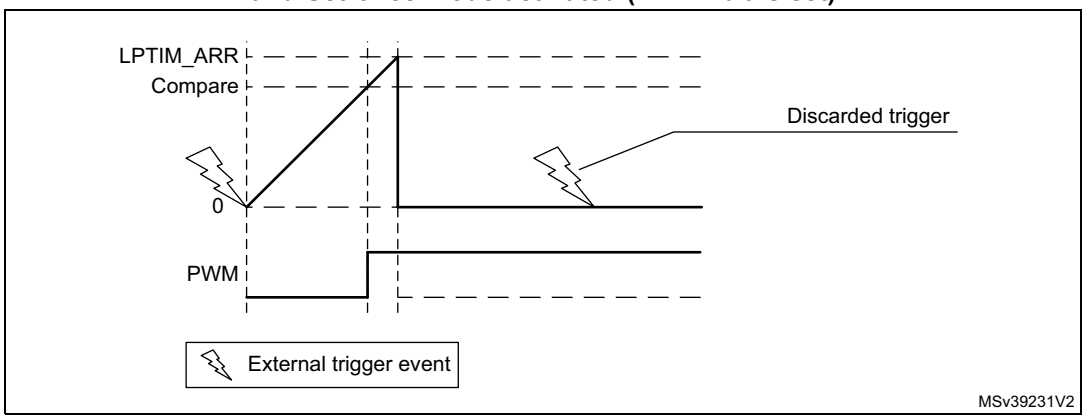
Figure 527. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 528*.

Figure 528. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

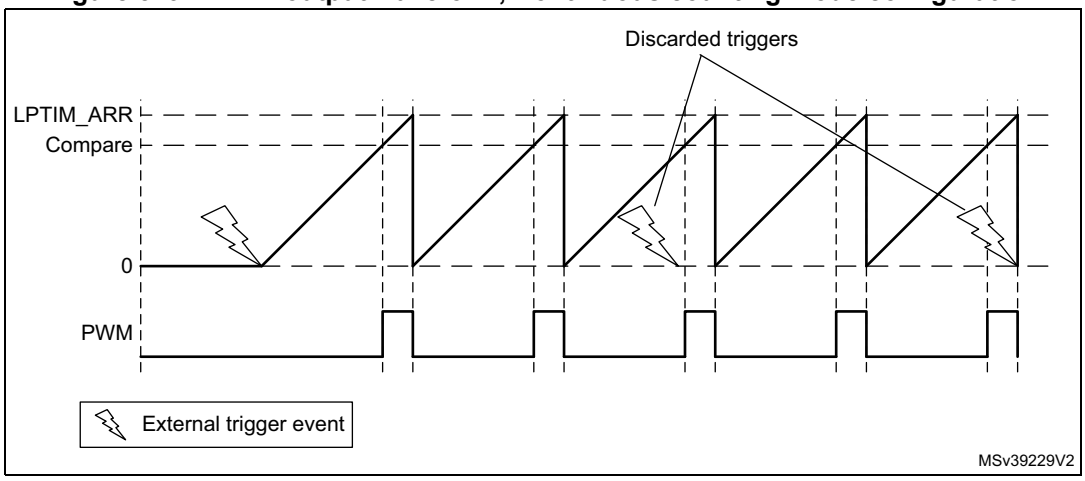
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 529*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 529. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

39.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

39.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 39.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

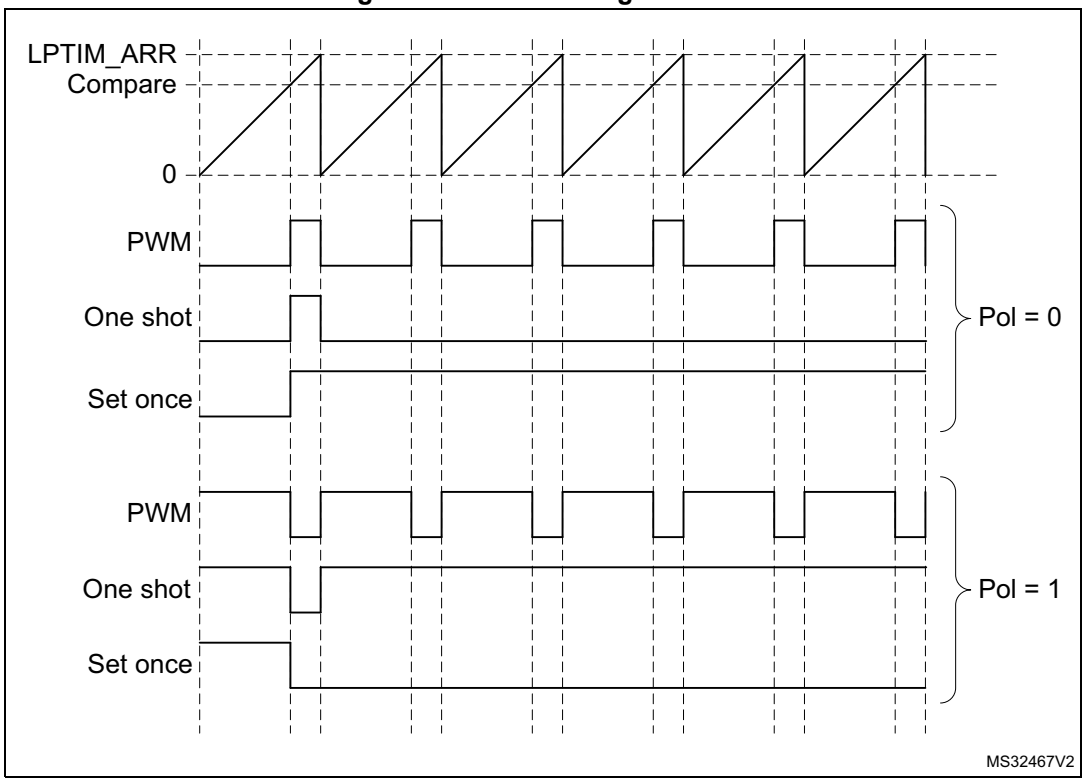
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 530* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

Figure 530. Waveform generation



39.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

39.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don't care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

39.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

39.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software should ensure that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

39.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

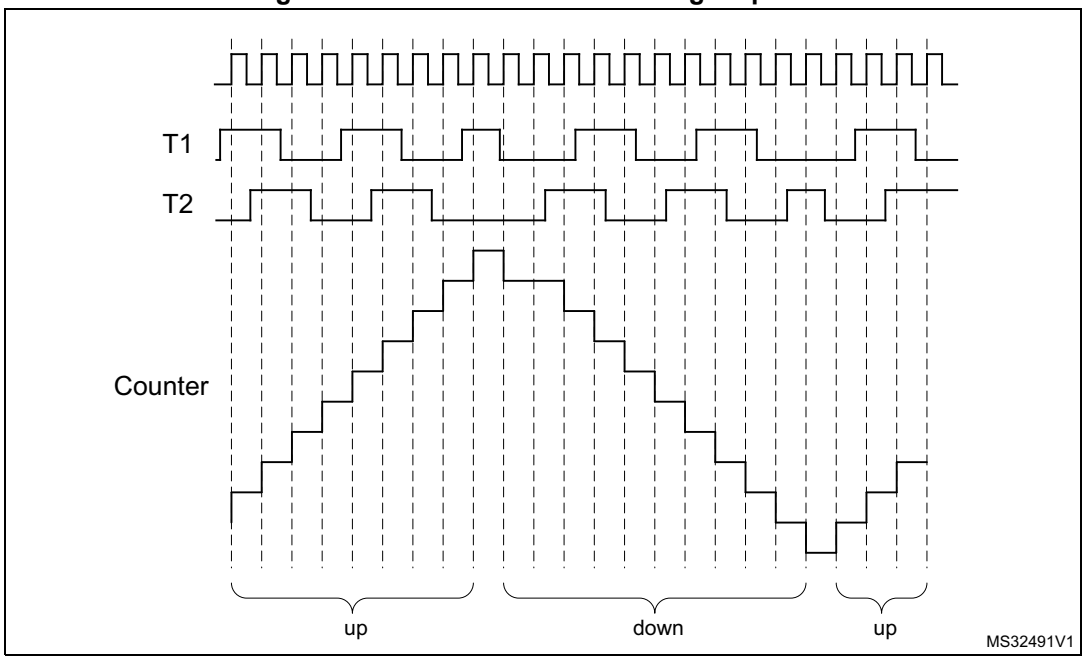
Table 420. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 531. Encoder mode counting sequence



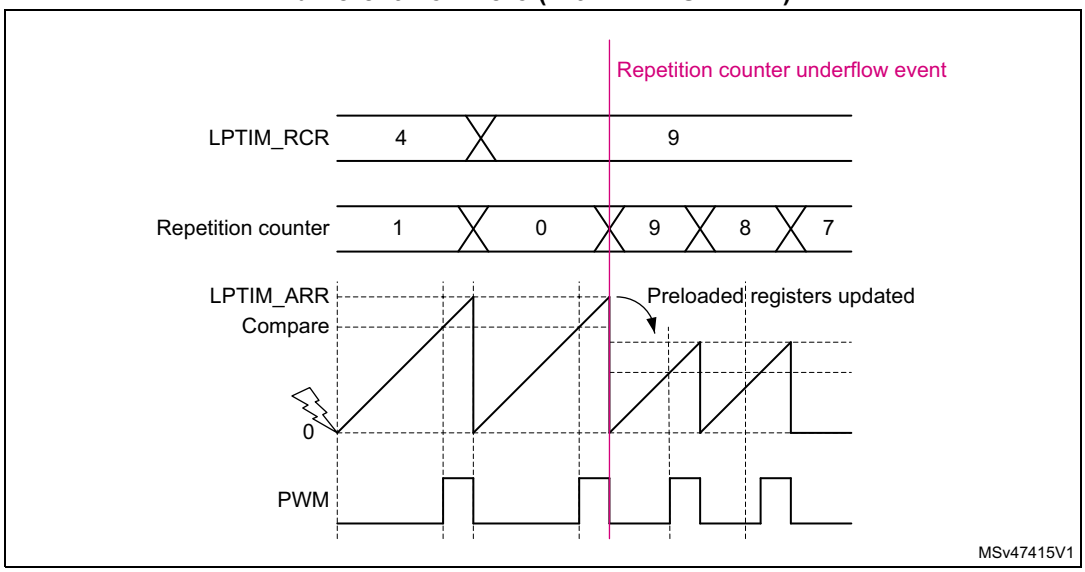
39.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 532*.

Figure 532. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 39.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

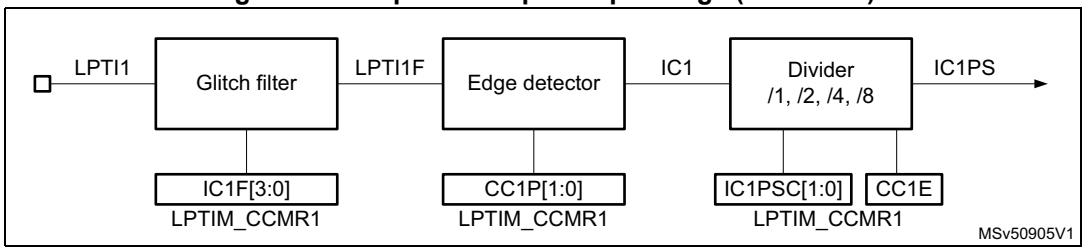
39.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

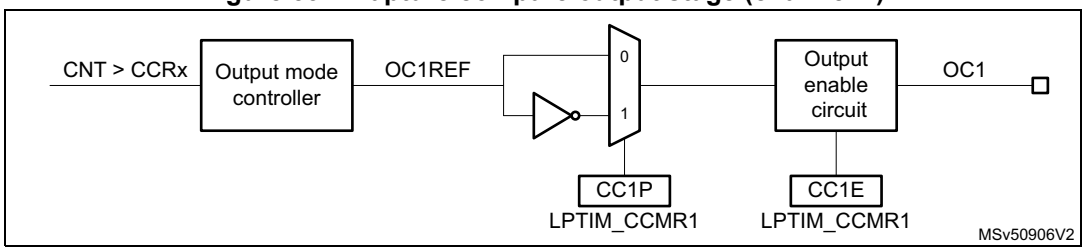
Figure 533. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 534. Capture/compare output stage (channel 1)



39.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 421*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 421*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 421. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 421. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

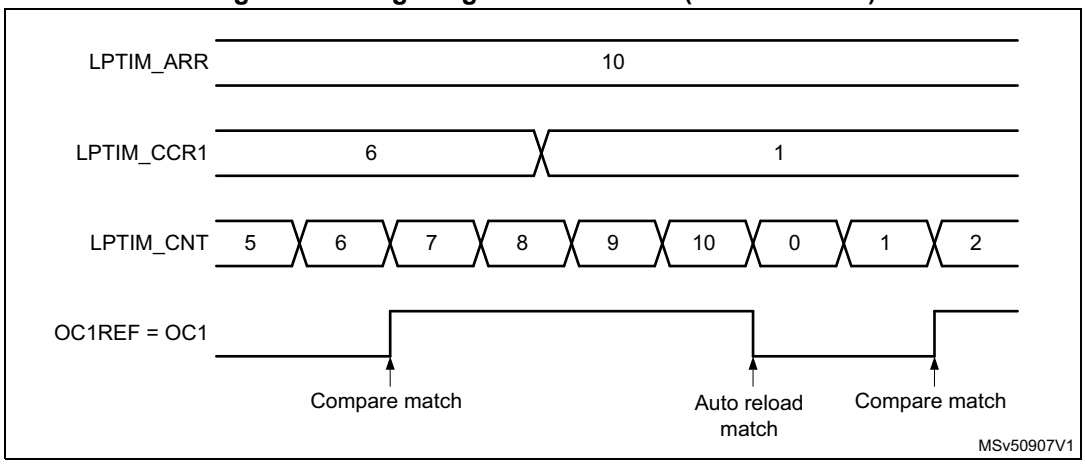
39.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 535 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

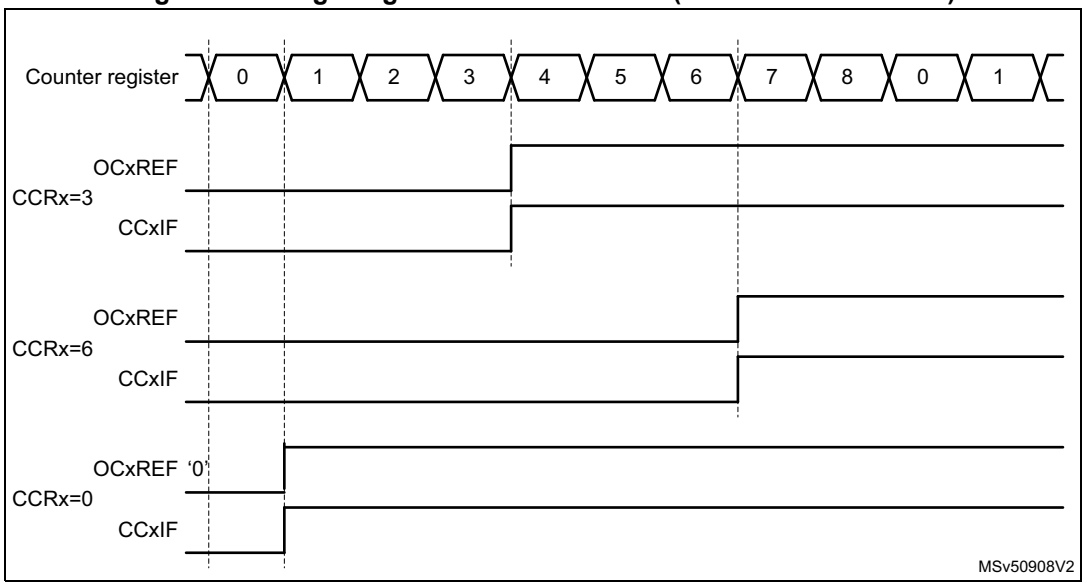
Figure 535. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 536 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 536. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

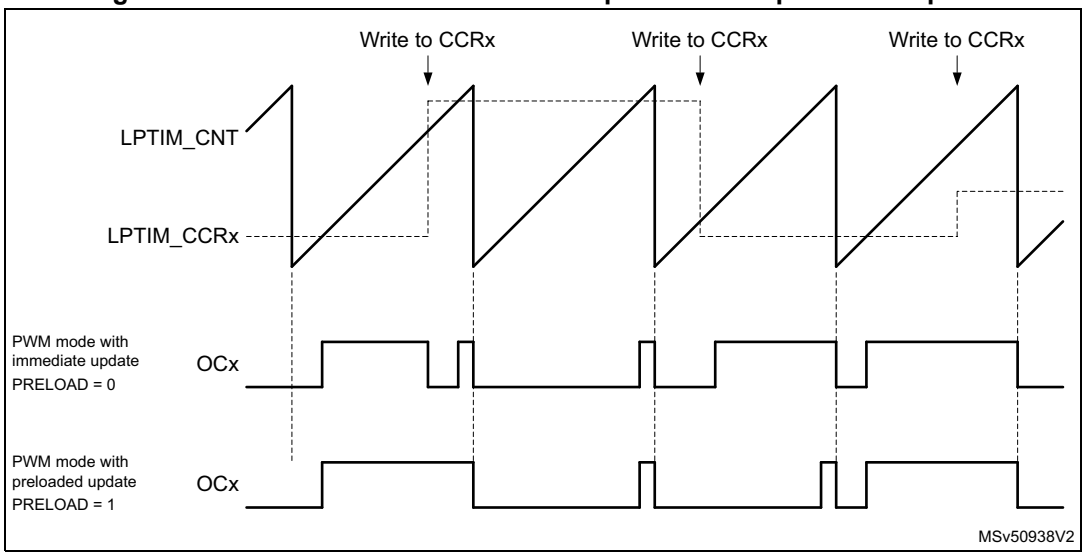
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 537* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 537. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 535 illustrates this behavior.

39.4.20 Autonomous mode

When clocked by oscillators available in this mode (refer to RCC), the LPTIM can operate in autonomous mode, remaining then fully functional in Stop mode where the APB clock is stopped. The APB clock is requested by the peripheral each time a data must be transferred from or to the SRAM. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the LPTIM configuration.

In order to offload the CPU (in Run mode) or to avoid to wake it up when in Stop mode, it is possible to use LPTIM DMA requests to transfer the captured values when in input capture mode or to update LPTIM registers when in PWM mode.

When in Stop mode, the LPTIM counter can be automatically started after the detection of an active edge on one of its external input triggers.

Input capture mode

To operate autonomously in Stop mode, the input capture DMA request must be enabled by setting the CCxDE bit in the LPTIM_DIER register.

Each time a counter value is captured and available in the LPTIM_CCRx register, the APB clock is requested by the peripheral and a DMA request is generated. The captured value is then transferred to the SRAM. The CCxIF flag is automatically cleared by hardware once the captured value is read by APB (can be any bus master like CPU or DMA).

PWM mode

The LPTIM can be configured to autonomously change, at each update event, the output waveform pulse width and/or the duty cycle without any CPU intervention. To enable this autonomous mode, the corresponding UEDE bit must be set in the LPTIM_DIER register.

At each update event, the APB clock is requested by the peripheral and a DMA request is generated. DMA direction must be configured as memory-to-peripheral which enables updating LPTIM registers, at each DMA request, with values stored in SRAM.

The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA. Thus, to enable automatic hardware clearing of UE flag, the application must configure the LPTIM_ARR register to be the last one to be written (at the end of list). For instance, if LPTIM_CCR1 and LPTIM_RCR registers need to be updated in Stop mode by DMA, the update sequence must be: LPTIM_CCR1, LPTIM_RCR then LPTIM_ARR.

The UE flag can also be cleared over its corresponding clear bit UECF in the LPTIM_ICR register, this can be done by configuring the DMA to write the LPTIM_ICR register at the end of register update.

39.4.21 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

39.4.22 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

39.5 LPTIM low-power modes

Table 422. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop(1)	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode. The DMA requests are functional if the instance supports the autonomous mode (refer to *Section 39.3: LPTIM implementation*).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Only LPTIM1 and LPTIM3 support autonomous mode with wake-up capability in Stop 2 mode. LPTIM4 does not support autonomous mode but has wake-up capability in stop 2 mode. LPTIM2 must be disabled before entering Stop 2 mode.

39.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 423. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 39.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 39.3: LPTIM implementation*.
3. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 39.3: LPTIM implementation*.

39.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 87* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

39.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

39.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

39.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OF	CC1 OF	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	Res.	EXT TRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

39.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OCF	CC1OCF	Res.	Res.	CC2CF	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

39.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 39.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 39.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

39.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 39.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers
Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 39.3.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: Software trigger (counting start is initiated by software)
01: Rising edge is the active edge
10: Falling edge is the active edge
11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 39.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger
01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition
01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 39.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

39.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

39.7.12 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

39.7.13 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

39.7.14 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value
When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

39.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 39.4.3: LPTIM input and trigger mapping*.

39.7.16 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

39.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 39.3.

39.7.18 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 39.3: LPTIM implementation*.

39.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 424. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM4_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0

Table 424. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM4_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0																0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM4_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIMx_DIER (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	WAVPOL(3)	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]		Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL	
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 424. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1(4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 39.3: LPTIM implementation*.
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 39.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 39.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 Infrared interface (IRTIM)

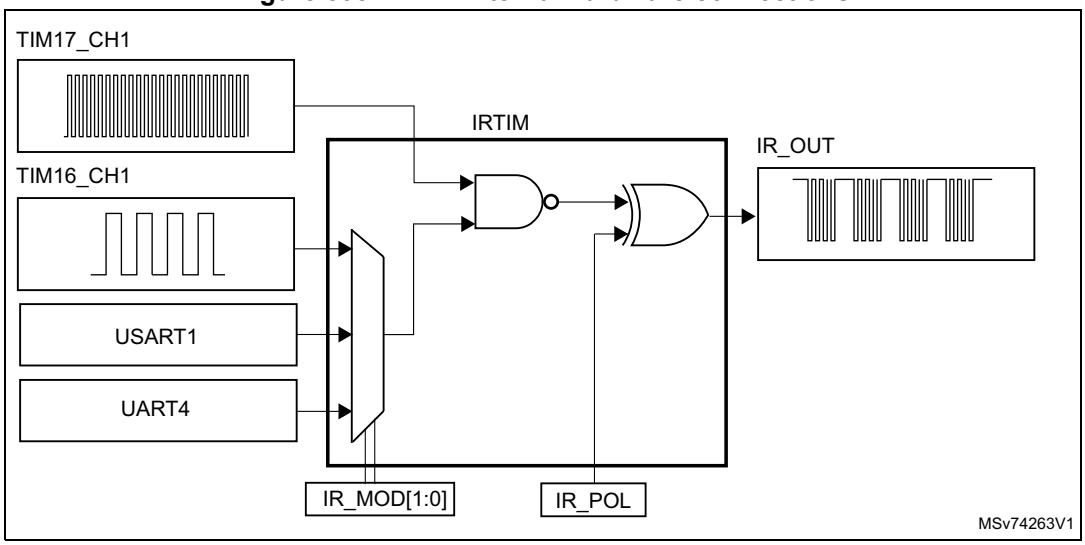
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with USART1, UART4, TIM16, and TIM17 as shown in *Figure 538*.

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16_OC1) and TIM17 channel 1 (TIM17_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

Figure 538. IRTIM internal hardware connections



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 or alternatively USART1 or UART4 generates the modulation envelope according to the setting of the IR_MOD[1:0] bits in the SYSCFG_CFGR1 register.

The polarity of the output signal from IRTIM is controlled by the IR_POL bit in the SYSCFG_CFGR1 register and can be inverted by setting of this bit.

The infrared function is output on the IR_OUT pin. The activation of this function is done through the GPIOx_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

41 Independent watchdog (IWDG)

41.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 41.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

41.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

41.3 IWDG implementation

41.4 IWDG functional description

Table 425. STM32U3 series IWDG features(1)

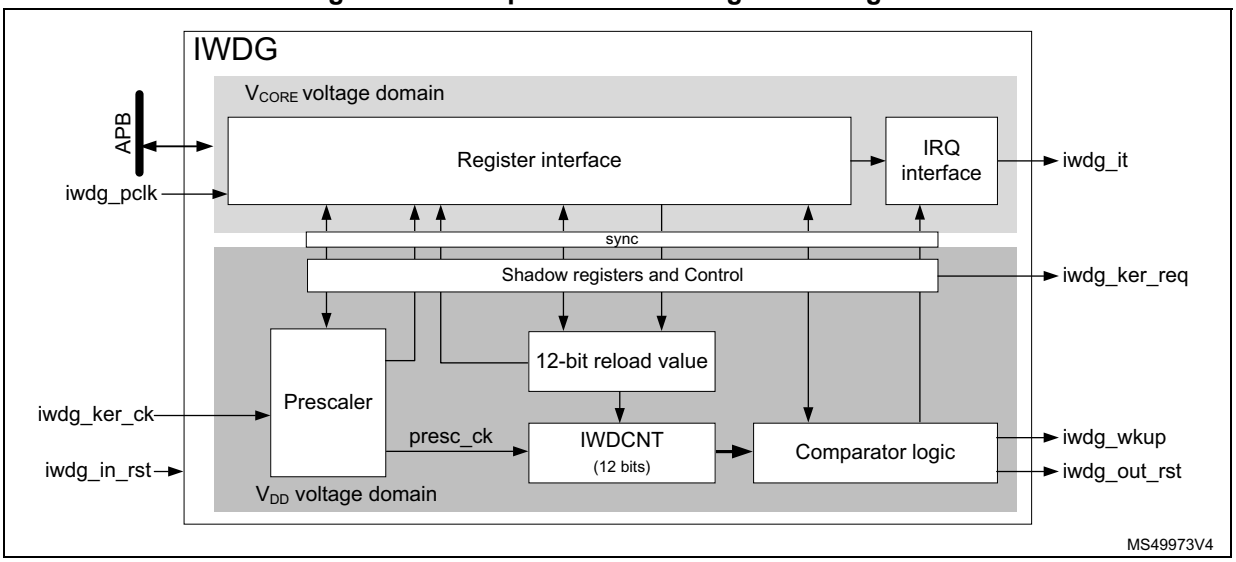
IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate a wake-up interrupt in system Stop(3)	X
Capability to generate a wake-up interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in debug mode(4)	X
Option bytes to control the activity in Stop mode(5)	X
Option bytes to control the activity in Standby mode(6)	X
Option bytes to control the hardware mode(7)	X

1. 'X' = supported, '-' = not supported.
2. Refer to the RCC section for additional information.
3. Wake-up from Stop with interrupt is supported only in Stop 0, Stop 1, and Stop 2 modes.
4. Controlled via DBG_IWDG_STOP in DBG section.
5. Controlled via the option byte IWDG_STOP in FLASH section.
6. Controlled via the option byte IWDG_STDBY in FLASH section.
7. Controlled via the option byte IWDG_SW in FLASH section.

41.4.1 IWDG block diagram

Figure 539 shows the functional blocks of the independent watchdog module.

Figure 539. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 41.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 426. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

41.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 427*.

Table 427. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request

Table 427. IWDG internal input/output signals (continued)

Signal name	Signal type	Description
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

41.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 41.3* for details.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

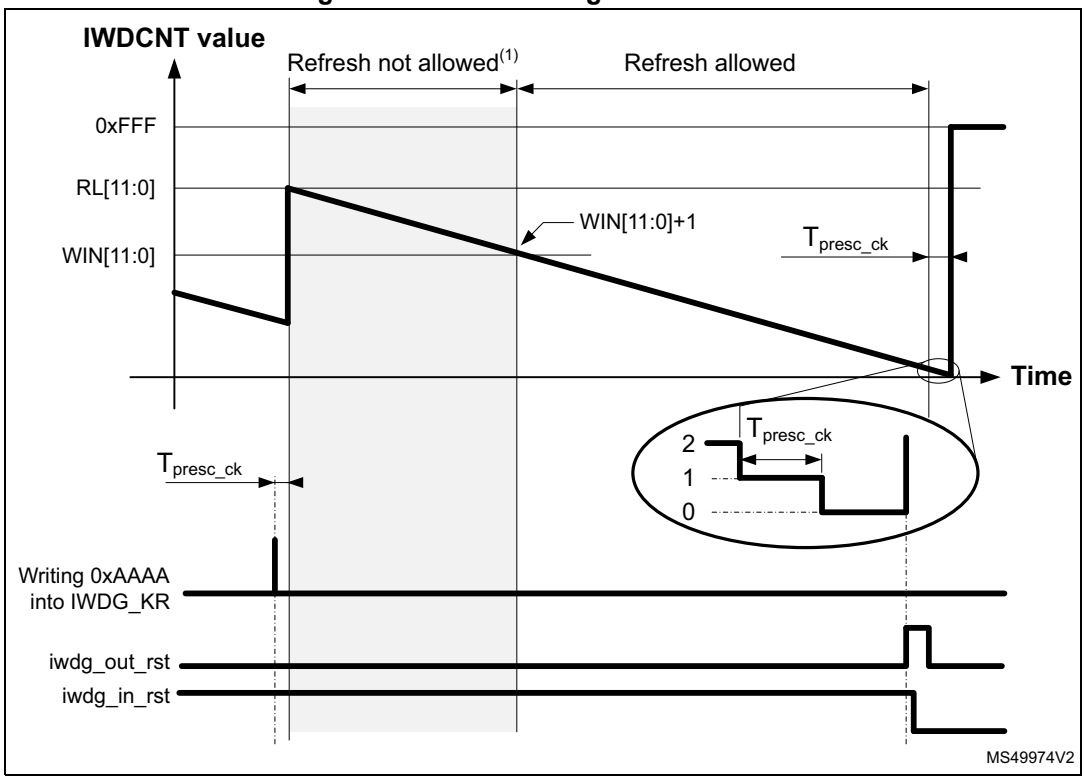
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 540*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 540. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

41.4.4 Window option

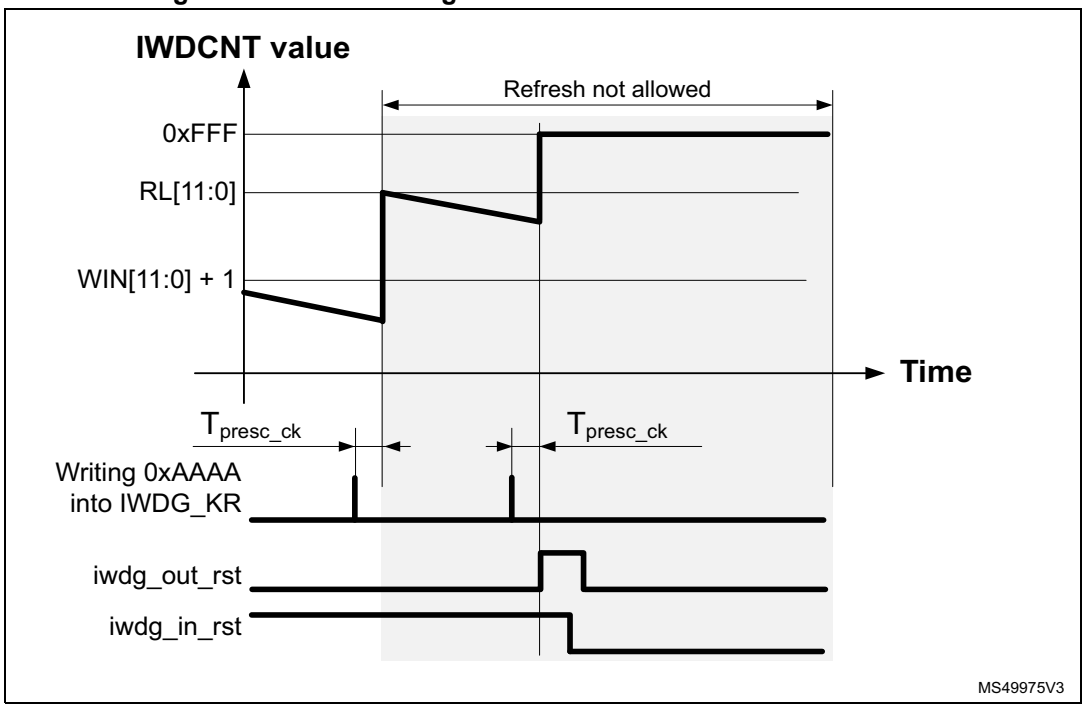
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 541*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 541. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

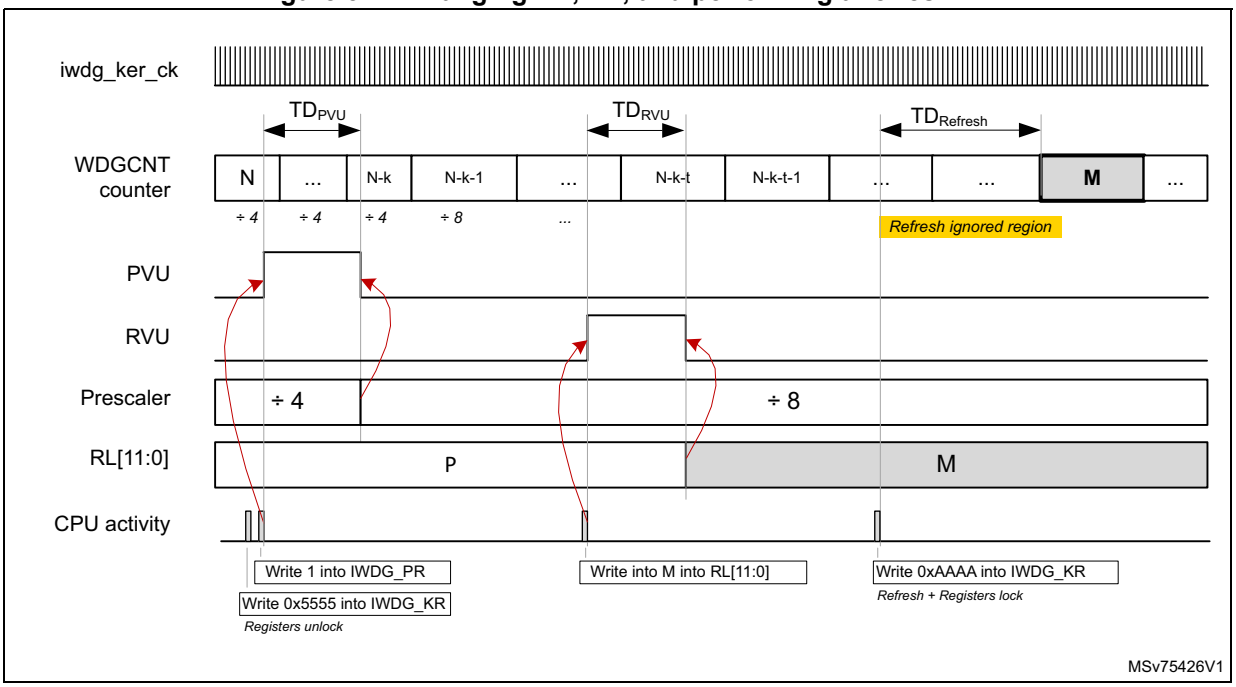
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 542. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 426: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

41.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 41.3* for details on the capabilities of this product.

41.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

41.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 41.3* for details.

Table 428. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 41.3* for details. IWDG interrupts cause the device exit the Stop mode.
Shutdown	The IWDG is not working.

41.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

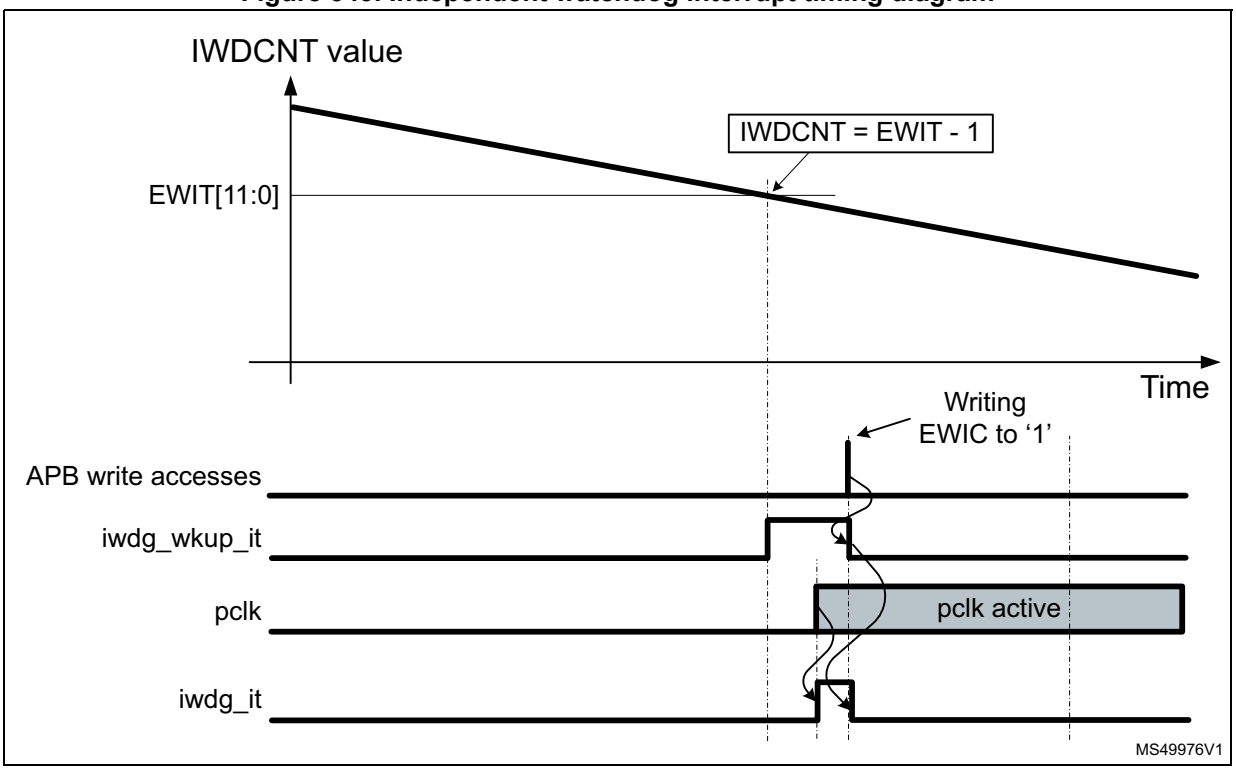
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG early wake-up interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 543. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.

2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Table 429 summarizes the IWDG interrupt request.

Table 429. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

41.7 IWDG registers

Refer to *Section 1.2 on page 87* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

41.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

41.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 41.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

41.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 41.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

41.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	r											r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 426: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 426: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 426: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 426: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

41.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 41.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

41.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	EWIC	Res.	Res.	EWIT[11:0]											
rw	w			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter window value

These bits are write access protected (see *Section 41.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

41.7.7 IWDG register map

Table 430. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0											0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

42 System window watchdog (WWDG)

42.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

42.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 545*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

42.3 WWDG implementation

Table 431. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

42.4 WWDG functional description

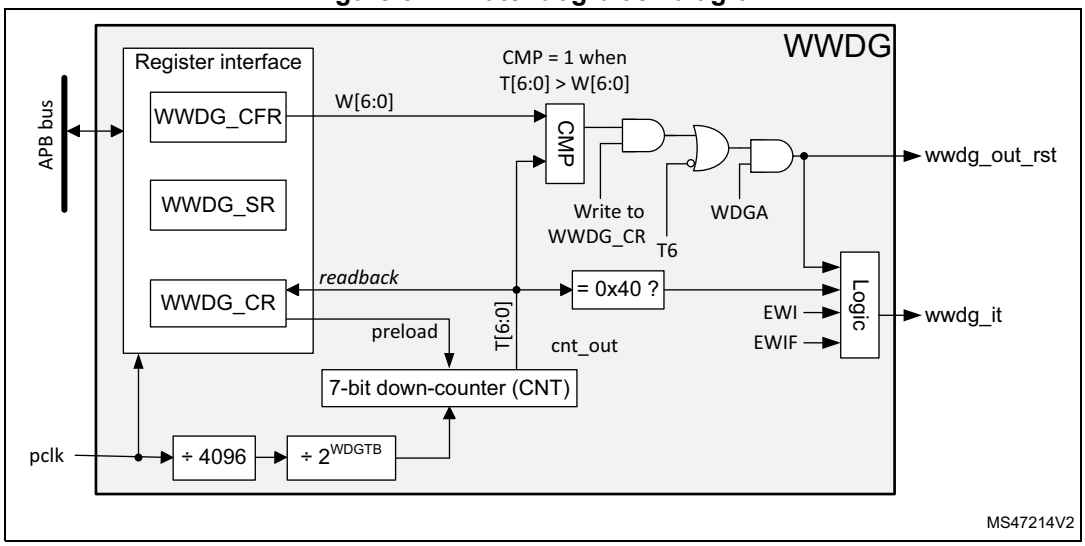
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 544* for the WWDG block diagram.

42.4.1 WWDG block diagram

Figure 544. Watchdog block diagram



42.4.2 WWDG internal signals

Table 432 gives the list of WWDG internal signals.

Table 432. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

42.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

42.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 545*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 545* describes the window watchdog process.

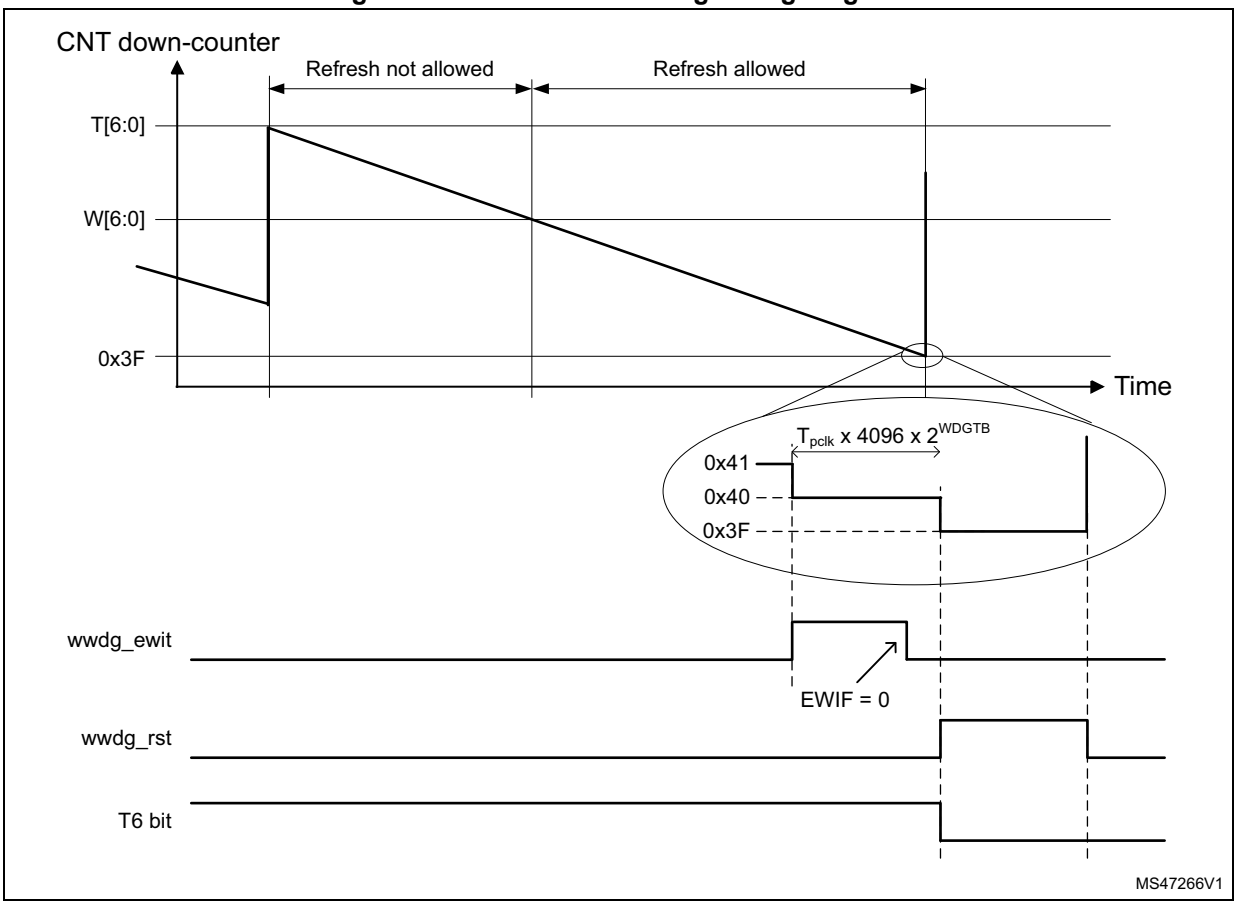
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

42.4.5 How to program the watchdog timeout

Use the formula in *Figure 545* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 545. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

42.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 53: Debug support (DBG)*.

42.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 433. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 42.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

42.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

42.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

42.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

42.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

42.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 434. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

43 Real-time clock (RTC)

43.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

43.2 RTC main features

The RTC supports the following features (see *Figure 546: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

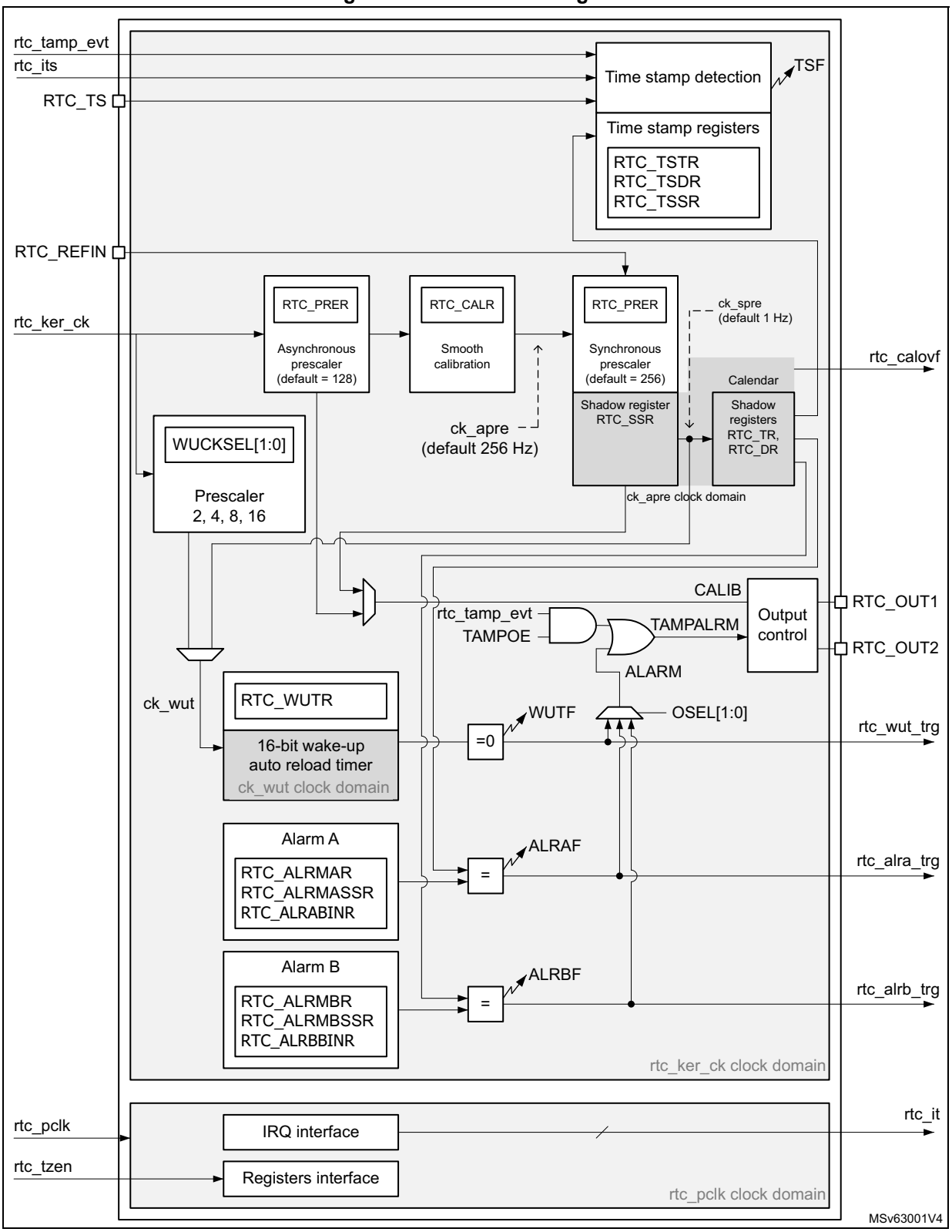
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

43.3 RTC functional description

43.3.1 RTC block diagram

Figure 546. RTC block diagram



43.3.2 RTC pins and internal signals

Table 435. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which selects one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 436. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 43.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 43.4: RTC low-power modes* for more details.

Table 437. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_tzen	From FLASH option bytes: TZEN
rtc_calovf	To TAMP peripheral: tamp_itamp5

The TZEN option bit is used to activate TrustZone in the device.

- TZEN = 1: TrustZone activated.
- TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB accesses to the RTC registers are nonsecure.

The triggers outputs can be used as triggers for other peripherals.

43.3.3 GPIOs controlled by the RTC

The GPIOs with RTC functions available in Standby mode are directly controlled by the RTC, whatever the GPIO configuration.

The different RTC functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, TAMPOE, COE and OUT2EN configuration, as shown in *Table 438*.

Table 438. RTC_OUT mapping

OSEL[1:0] bits	TAMPOE	COE bit	OUT2EN bit	RTC_OUT1	RTC_OUT2
00	0	0	0	-	-
		1		CALIB	-
Don't care	1	Don't care		TAMPALRM	-
01 or 10 or 11	Don't care				-
00	0	0	1	-	-
		1		-	CALIB
Don't care	1	0		-	TAMPALRM
01 or 10 or 11	Don't care			-	
Don't care	1	1		TAMPALRM	CALIB
01 or 10 or 11	Don't care				

43.3.4 RTC secure protection modes

By default after a Backup domain reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR, RTC_TSSSR, RTC_TAMPTSCR and RTC_TSIDR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection: a notification is generated through a flag/interrupt in the IAC (illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

43.3.5 RTC privilege protection modes

By default after a Backup domain reset, all RTC registers can be read or written in both privileged and non-privileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and non-privilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in

RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.

  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR, RTC_TSSSR, RTC_TAMPTSCR and RTC_TSIDR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

43.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 546: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 43.3.10: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

43.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 43.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

43.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

43.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

43.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 1717*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 43.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

43.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCFGR, and/or SEC or INITSEC is set in the RTC_SECCFGR: the INIT KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR, and/or SEC or CALSEC is set in the RTC_SECCFGR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   - If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   - If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   - If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   - If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   – If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   – If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   – If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   – If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

43.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1716*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 43.3.14: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

43.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

43.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

43.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

43.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are

set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

43.3.17 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS or any bit of RTC_TAMPTSCR is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and *RTC timestamp on tamper control register (RTC_TAMPTSCR)*.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register. The TSID bitfield in the RTC_TSIDR provides the identification of the cause of the timestamp event. It is mandatory to check ITSF after clearing TSF status flag, in case an internal event occurs after first timestamp event. If set, ITSF must also be cleared by software to allow next timestamp event correct identification.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags. TSID indicates the source corresponding to time stored in timestamp registers, but due to this difference of timing and priority between events, RTC_TS or rtc_its can be identified as first event, even if it occurs up to 1 ck_apre after TAMP event.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

43.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

43.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

43.4 RTC low-power modes

Table 439. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.
Shutdown	The RTC remains active when the RTC clock source is LSE. RTC interrupts cause the device to exit the Shutdown mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 440. RTC pins functionality over modes

Functions	Functional in all low-power modes except Stop 3, Standby and Shutdown modes	Functional in Stop 3, Standby and Shutdown modes	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	Yes	No

43.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

Table 441. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 442. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

43.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

43.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1716* and *Reading the calendar on page 1718*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be write-protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

43.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1716* and *Reading the calendar on page 1718*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be write-protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

43.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

43.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wake-up timer configuration update not allowed except in initialization mode

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

43.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 1716*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be write-protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

43.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

43.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1716.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.

1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM

1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved if none of the RTC_TAMPSCR bit is set

1: Save RTC timestamp on tamper detection event (for any tamper event, whatever RTC_TAMPSCR value)

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled

1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.

00: Output disabled

01: Alarm A output enabled

10: Alarm B output enabled

11: Wake-up output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.

0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 43.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC_REFIN detection disabled

1: RTC_REFIN detection enabled

Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

0: RTC_TS input rising edge generates a timestamp event

1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.

11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

43.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 43.3.5: RTC privilege protection modes*).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or non-privileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or non-privileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, Delight saving, calibration and reference clock privilege protection

0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.

1: Shift register, Delight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 43.3.5: RTC privilege protection modes for details on the read protection.

43.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 43.3.5: RTC privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection
0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.
1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection
0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection
0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection
0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 43.3.4: RTC secure protection modes for details on the read protection.

43.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

43.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be write-protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 43.3.16: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 43.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 43.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 43.3.16: RTC smooth digital calibration on page 1721*.

43.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1716*.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

43.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

43.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

43.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

43.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

43.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

43.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

43.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

43.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

43.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSR UMF	ITS MF	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag
This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp nonsecure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp nonsecure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

43.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSR UMF	ITS MF	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

43.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

43.6.24 RTC timestamp on tamper control register (RTC_TAMPTSCR)

Address offset: 0x64

Backup domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMPTS
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5TS	TAMP4TS	TAMP3TS	TAMP2TS	TAMP1TS
											rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ITAMPTS**: Timestamp on internal tamper event
- 0: When TAMPTS = 0, internal tamper detection event does not cause a RTC timestamp to be saved
- 1: Save RTC timestamp on any internal tamper detection event (whatever TAMPTS value)

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TAMP5TS**: Timestamp on external tamper TAMP5 event
- 0: When TAMPTS = 0, TAMP5 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP5 detection event (whatever TAMPTS value)

Bit 3 **TAMP4TS**: Timestamp on external tamper TAMP4 event
- 0: When TAMPTS = 0, TAMP4 detection event does not cause an RTC timestamp to be saved
- 1: Save RTC timestamp on TAMP4 detection event (whatever TAMPTS value)

Bit 2 **TAMP3TS**: Timestamp on external tamper TAMP3 event
0: When TAMPTS = 0, TAMP3 detection event does not cause an RTC timestamp to be saved
1: Save RTC timestamp on TAMP3 detection event (whatever TAMPTS value)

Bit 1 **TAMP2TS**: Timestamp on external tamper TAMP2 event
0: When TAMPTS = 0, TAMP2 detection event does not cause an RTC timestamp to be saved
1: Save RTC timestamp on TAMP2 detection event (whatever TAMPTS value)

Bit 0 **TAMP1TS**: Timestamp on external tamper TAMP1 event
0: When TAMPTS = 0, TAMP1 detection event does not cause an RTC timestamp to be saved
1: Save RTC timestamp on TAMP1 detection event (whatever TAMPTS value)

Note: All bits of RTC_TAMPTSCR must be cleared before entering RTC initialization mode.

43.6.25 RTC timestamp status register (RTC_TSIDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR.

Address offset: 0x68

Backup domain reset value: 0x0000 0000

System reset: not affected

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSID[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TSID[5:0]**: Timestamp flag source identification
This bitfield is updated with identification of the source generating the timestamp stored in timestamp registers. It is updated at each TSF event, and is not affected by TSOVF event.
000000: RTC_TS pin event
000001: TAMP1F event
000010: TAMP2F event
000011: TAMP3F event
000100: TAMP4F event
000101: TAMP5F event
010000: Internal tamper event
100000: Internal timestamp event (from rtc_its internal signal)
Others: Reserved

43.6.26 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

43.6.27 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 43.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 43.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

43.6.28 RTC register map

Table 443. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 443. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																		0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																									0	0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																									0	0	0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																									0	0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																									0	0	0	0	0	0	0	0
0x64	RTC_ TAMPTSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMPTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5TS	TAMP4TS	TAMP3TS	TAMP2TS	TAMP1TS
	Reset value																0												0	0	0	0	0
0x68	RTC_TSIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSID[5:0]					
	Reset value																											0	0	0	0	0	0

Table 443. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

44 Tamper and backup registers (TAMP)

44.1 Introduction

The antitamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this antitamper detection circuit with five tamper pins and nine internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

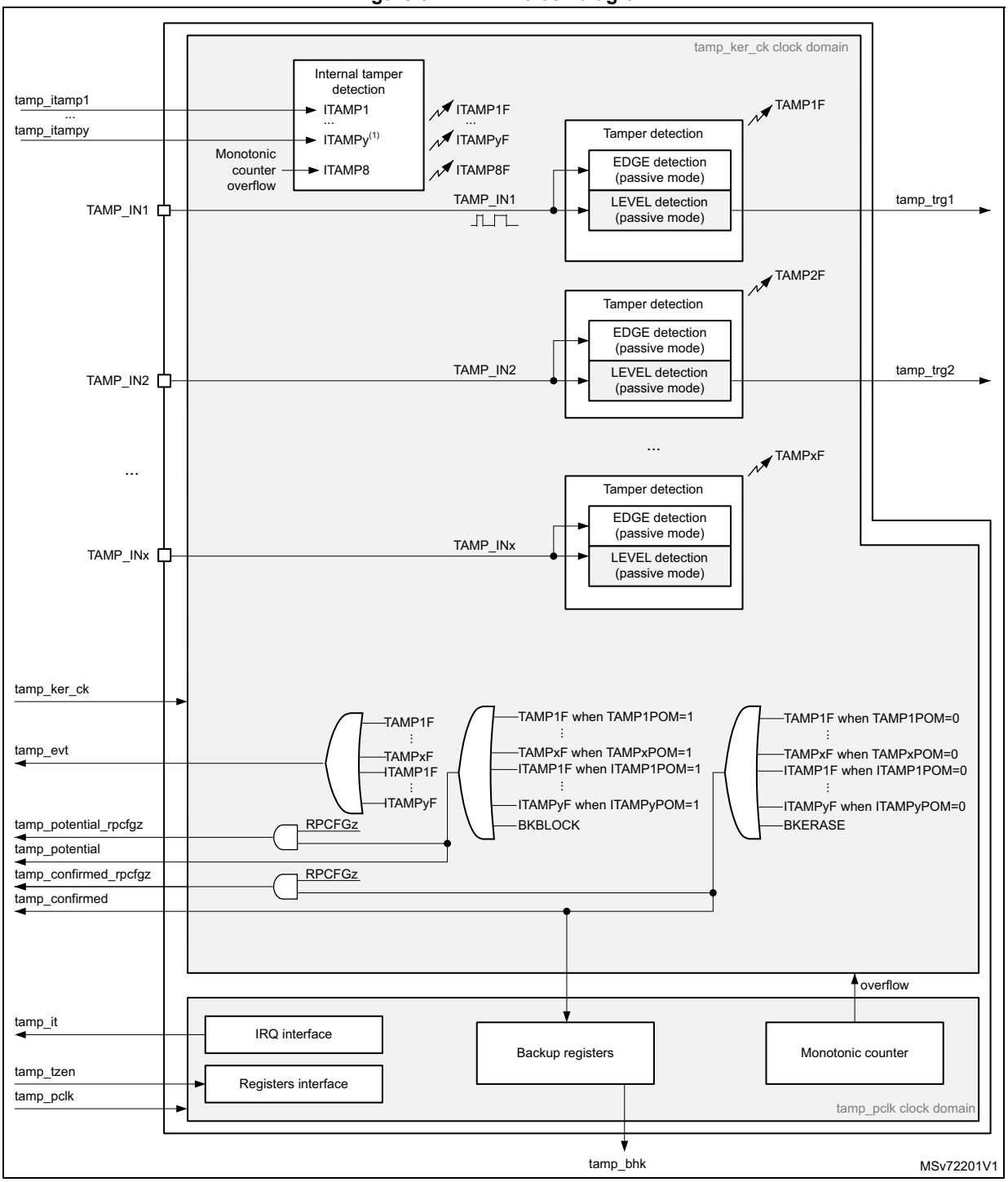
44.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAM2, cache, and cryptographic peripherals. The device resources protected by a tamper are named "device secrets". The list of device secrets is configurable by software.
- 32 32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Five tamper pin for five external tamper detection event:
  - Passive tampers: ultralow power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- Nine internal tamper events to protect against transient or environmental perturbation attacks.
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase.
  - Potential mode: most of the secrets erase following a tamper detection are launched by software.
- Any tamper detection can generate an RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in three configurable-size areas:
    1 read/write secure area
    1 write secure/read nonsecure area
    1 read/write nonsecure area.
- Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter

44.3 TAMP functional description

44.3.1 TAMP block diagram

Figure 547. TAMP block diagram



1. The number of external and internal tampers depends on products.

44.3.2 TAMP pins and internal signals

Table 444. TAMP input/output pins

Pin name	Pin type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin

Table 445. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets protection(1). This signal is active when: a tamper event detection flag (internal or external tamper), is generated in potential mode. or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) ptotection. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in confirmed mode. or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in potential mode. or a software request is done by writing BKBLOCK to 1
tamp_confirmed_ rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: a tamper event detection flag (internal or external tamper), is generated in confirmed mode. or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 44.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

1. Refer to *Table 446: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to *Section 44.4: TAMP in low-power modes* for more details.

Table 446. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	From FLASH option bytes: TZEN
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	No effect
tamp_confirmed	No effect
tamp_potential_ rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg1 signal is used to block the read and write accesses to the device secrets listed hereafter: – SRAM2
tamp_confirmed_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg1 signal is used to erase the device secrets listed hereafter: – SRAM2 The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg2 signal is used to block the device secrets listed hereafter: – RHUK and to erase the device secrets listed hereafter: – SAES, AES, CCB
tamp_confirmed_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg2 signal is used to block the device secrets listed hereafter: – RHUK and to erase the device secrets listed hereafter: – SAES, AES, CCB
tamp_potential_rpcfg3	When the bit RPCFG3 is set in the TAMP_RPCFGR, the tamp_potential_ rpcfg3 signal is used to erase the device secrets listed hereafter: – ICACHE The device secrets access is blocked when erase is ongoing.
tamp_confirmed_rpcfg3	When the bit RPCFG3 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg3 signal is used to erase the device secrets listed hereafter: – ICACHE The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg4 signal is used to erase the device secrets listed hereafter: – HASH

Table 446. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_confirmed_rpcfg4	When the bit RPCFG4 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg4 signal is used to erase the device secrets listed hereafter: – HASH
tamp_potential_rpcfg5	When the bit RPCFG5 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg5 signal is used to erase the device secrets listed hereafter: – PKA including PKA SRAM The device secrets access is blocked when erase is ongoing.
tamp_confirmed_rpcfg5	When the bit RPCFG5 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg5 signal is used to erase the device secrets listed hereafter: – PKA including PKA SRAM The device secrets access is blocked when erase is ongoing.
tamp_itamp3	LSE monitoring (LSECSS)(1)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access when RDP > 0
tamp_itamp7	ADC1 (adc1_awd1) analog watchdog monitoring 1
tamp_itamp8(2)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (SAES or AES or PKA or TRNG or CCB)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC1 (adc1_awd2) analog watchdog monitoring 2
tamp_itamp13	ADC1 (adc1_awd3) analog watchdog monitoring 3
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. This monitoring must be enabled by setting LSECSSON in *RCC backup domain control register (RCC_BDCR)*.
2. This signal is generated in the TAMP peripheral.

The TZEN option bit is used to activate TrustZone in the device.

- TZEN = 1: TrustZone activated.
- TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB accesses to the TAMP registers are nonsecure.

44.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs with TAMP functions available in Standby mode are directly controlled by the TAMP, whatever the GPIO configuration.

The TAMP_INy function is enabled on a GPIO when TAMPyE = 1.

The GPIOs included in the battery backup domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS and TAMP_IN1 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

Table 447. PC13 pin configuration(1)

Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	0	0	Don't care	Don't care
		00	1					
		01 or 10 or 11	1					
TAMPALRM(2) Open-Drain	No pull	01 or 10 or 11	0	Don't care	1	0	Don't care	Don't care
		00	1					
		01 or 10 or 11	1					
	Internal pull-up	01 or 10 or 11	0			1		
		00	1					
		01 or 10 or 11	1					
CALIB output PP		00	0	1	Don't care	Don't care	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	1	0
		00	0	1				
		Don't care	Don't care	0				
RTC_TS and TAMP_IN1 input floating		00	0	0	Don't care	Don't care	1	1
		00	0	1				
		Don't care	Don't care	0				
RTC_TS input floating		00	0	0	Don't care	Don't care	0	1
		00	0	1				
		Don't care	Don't care	0				
Wake-up pin or Standard GPIO		00	0	0	Don't care	Don't care	0	0
		00	0	1				
		Don't care	Don't care	0				

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

44.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

44.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register (TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers and monotonic counter which have their own protection setting.
- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode.

A nonsecure access to a secure-protected register is denied:

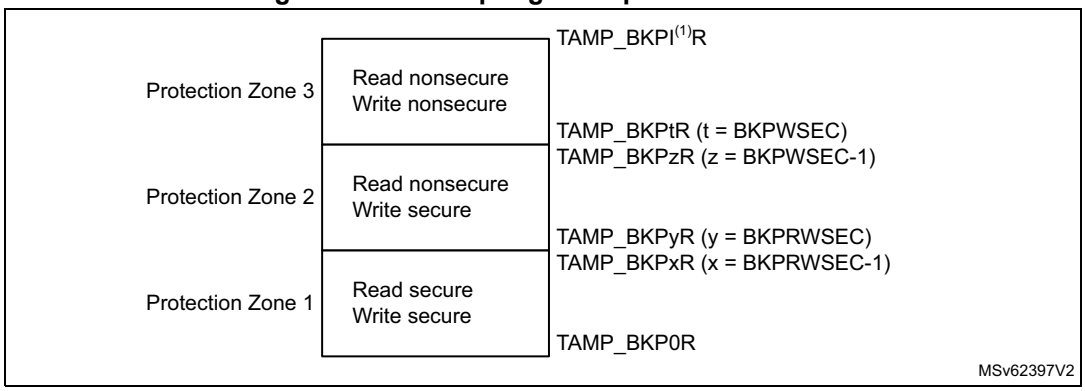
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the IAC (illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

44.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 548. Backup registers protection zones



1. I = last backup register index

In case TZEN =1, the bits BKPWPRIV and BKPRWPRIV in the TAMP_PRIVCFGR can be written only in secure mode.

44.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and nonprivileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:

- Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- Reading TAMP_SECCFGR, TAMP_PRIVCFGR is always possible in privilege and nonprivilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counter which have their own protection setting.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0], and BKPWPRIV for the protection zone 2 (refer to *Figure 548: Backup registers protection zones*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A nonprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

44.3.8 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the protection zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the eight backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a confirmed tamper event (if RPCFG31 = 1) or when the readout protection (RDP) is disabled. It is also cleared with BKERASE command if RPCFG31 = 1 (in all cases the backup registers are also erased).

Refer to section secure AES coprocessor (SAES) for details on the procedure to download the boot hardware key in the SAES.

44.3.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- Erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 446: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- Block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 446: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- Generate an interrupt, capable to wake up from low-power modes.
- Generate a hardware trigger for the low-power timers, or an RTC timestamp event.

The external I/Os tamper detection supports passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or level.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

44.3.10 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set

(TAMPxMSK = 1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.

44.3.11 Tamper detection - confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 446: TAMP interconnection*.

This mode is selected to automatically erase the device secrets when the tamper is detected.

Tamper detection - potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in *Table 446: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to

launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirm tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 446: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the "potential tamper" into "confirmed tamper" if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode).
- It is not affected by BKERASE software command.
- It is not affected by BKBLOCK software command.

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 446: TAMP interconnection*.
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 446: TAMP interconnection*.

Table 448. Backup registers and BHK tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFG31 = 0	No effect on backup registers, BHK and BHKLOCK	No effect on backup registers, BHK and BHKLOCK
RPCFG31 = 1	Backup registers and BHK access is blocked	Backup registers, BHK and BHKLOCK are erased

Table 449. Other device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_ rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 446: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

44.3.12 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register. Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as a trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00). Refer to *Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. When TAMPTS is 0, any external tamper can be individually configured to cause or not a timestamp. A global configuration for all internal tampers is also available to enable or not timestamp on internal tamper detection (refer to RTC_TAMPTSCR). In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs. The TSID bitfield in the RTC_TSIDR) provides the identification of the cause of the timestamp event.

Caution: Timestamp on tamper event must be disabled before entering RTC initialization mode (TAMPTS, and all bits of RTC_TAMPTSCR must be cleared before setting the INIT bit in RTC_ICSR).

Note: TSF is set up to three ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1 or any bit in TAMP_TAMPTSCR register is set, the software either waits for the timestamp flag before clearing the tamper flag, or reads the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then reenabled (TAMPxE set to 1) before reprogramming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a nonzero value. A tamper detection event is generated when either two, four, or eight (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 549. Tamper sampling with precharge pulse

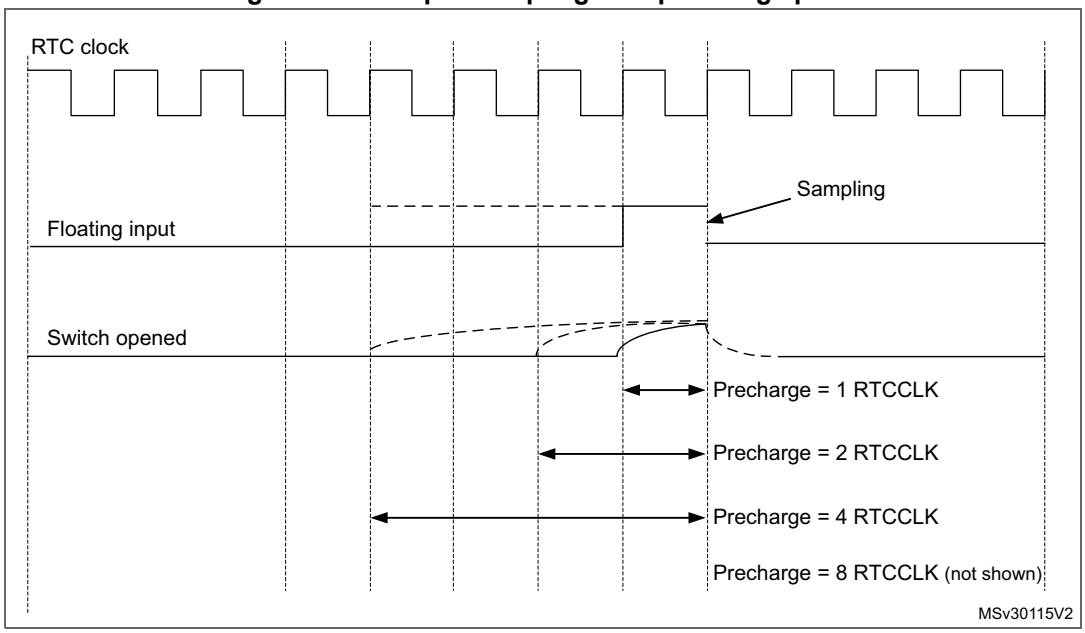
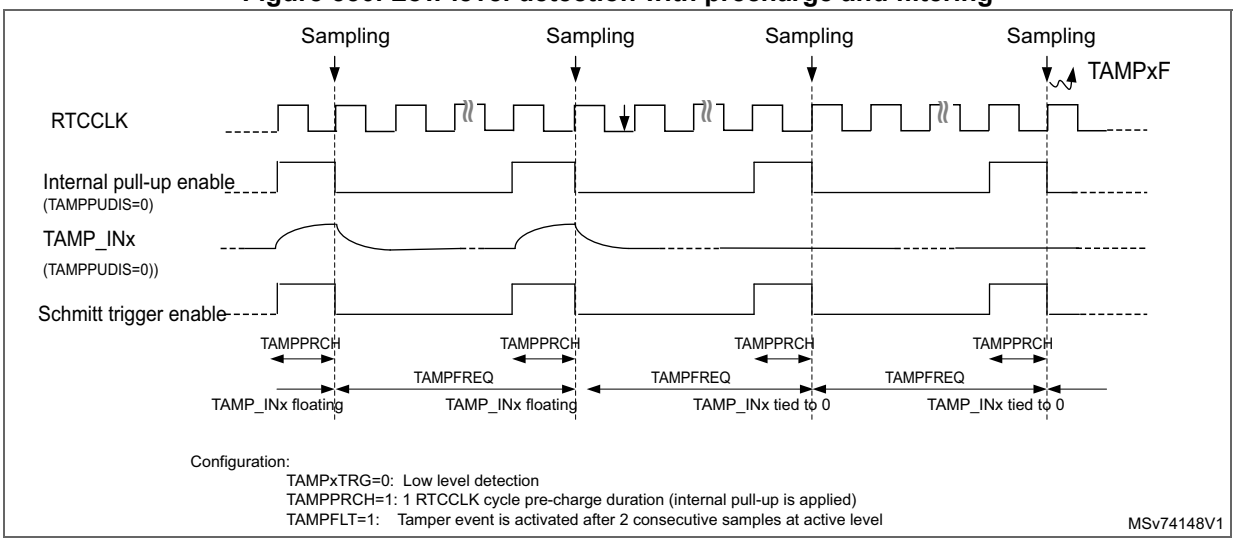


Figure 550. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

44.4 TAMP in low-power modes

Table 450. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. Tamper interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. Tamper interrupts cause the device to exit the Standby mode.
Shutdown	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE. Tamper interrupts cause the device to exit the Shutdown mode.

Table 451. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[5:1]	Yes	Yes

44.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 452. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Section 44.4: TAMP in low-power modes* for more details about available features in the low-power modes.

44.6 TAMP registers

44.6.1 TAMP control register 1 (TAMP_CR1)

Address offset: 0x000

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*. This register can be protected against unprivileged access (refer to*Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
											rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ITAMPiE**: Internal tamper i enable (i = 13 to 11)
0: Internal tamper i disabled
1: Internal tamper i enabled

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **ITAMPiE**: Internal tamper i enable (i = 9 to 5)
0: Internal tamper i disabled
1: Internal tamper i enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled
1: Internal tamper 3 enabled

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiE**: Tamper detection on TAMP_INi enable (i = 5 to 1)
0: Tamper detection on TAMP_INi disabled
1: Tamper detection on TAMP_INi enabled

Note: Tamper detection mode (selected with TAMP_FLTCR and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

44.6.2 TAMP control register 2 (TAMP_CR2)

Address offset: 0x004

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BKERA SE	BKBLO CK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
			rw	rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5 POM	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
											rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **TAMPiTRG**: Active level for tamper 5 input (i = 5 to 1)

If TAMPFLT[1:0] = 00 in TAMP_FLTCR, the tamper i input rising edge triggers a tamper detection event, and the tamper i input falling edge triggers a tamper detection event.

0: If TAMPFLT[1:0] ≠ 00, the tamper i input staying low triggers a tamper detection event.

1: If TAMPFLT[1:0] ≠00 the tamper i input staying high triggers a tamper detection event.

Bit 23 **BKERASE**: Software confirmed tamper

Writing 1 to this bit raises the tamp_confirmed and tamp_confirmed_rpcfgx signals. Writing 0 has no effect. This bit is always read as 0.

Note: The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 446: TAMP interconnection.

Bit 22 **BKBLOCK**: Software potential tamper

Writing 1 to this bit raises the tamp_potential and tamp_potential_rpcfgx signals.

0: Backup registers and device secrets can be accessed if no tamper flag is set.

1: Backup registers and device secrets cannot be accessed.

Note: The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 446: TAMP interconnection.

Bits 21:19 Reserved, must be kept at reset value.

Bits 18:16 **TAMPiMSK**: Tamper i mask (i = 3 to 1)

0: Tamper i event generates a trigger event and TAMPiF must be cleared by software to allow next tamper event detection.

1: Tamper i event generates a trigger event. TAMPiF is masked and internally cleared by hardware. Backup registers and device secrets are not erased.

Note: The tamper i interrupt must not be enabled when TAMPiMSK is set.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiPOM**: Tamper i potential mode (i = 5 to 1)

0: Tamper i event detection is in confirmed mode.

1: Tamper i event detection is in potential mode.

Note: The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 446: TAMP interconnection. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 446: TAMP interconnection.

44.6.3 TAMP control register 3 (TAMP_CR3)

Address offset: 0x008

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	Res.	ITAMP3POM	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:10 **ITAMPiPOM**: Internal tamper i potential mode (i = 13 to 11)

0: Internal tamper i event detection is in confirmed mode.

1: Internal tamper i event detection is in potential mode.

Note: The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 446: TAMP interconnection. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 446: TAMP interconnection.

Bit 9 Reserved, must be kept at reset value.

Bits 8:4 **ITAMPiPOM**: Internal tamper i potential mode (i = 9 to 5)

0: Internal tamper i event detection is in confirmed mode.

1: Internal tamper i event detection is in potential mode.

Note: The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 446: TAMP interconnection. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 446: TAMP interconnection.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode

0: Internal tamper 3 event detection is in confirmed mode.

1: Internal tamper 3 event detection is in potential mode.

Note: The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 446: TAMP interconnection. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 446: TAMP interconnection.

Bits 1:0 Reserved, must be kept at reset value.

44.6.4 TAMP filter control register (TAMP_FLTCR)

Address offset: 0x00C

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each TAMPx pin is precharged before each sample.

0: Precharges TAMP_INx pins before sampling (enable internal pull-up)

1: Disables precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**:TAMP_INx precharge duration

This bitfield determines the time duration during which the pull-up is activated before each sample. TAMPPRCH is valid for each TAMP_INx input.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**:TAMP_INx filter count

This bitfield determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each TAMP_INx input.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after two consecutive samples at the active level.

0x2: Tamper event is activated after four consecutive samples at the active level.

0x3: Tamper event is activated after eight consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

This bitfield determines the frequency at which each TAMP_INx input is sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

44.6.5 TAMP secure configuration register (TAMP_SECCFGR)

Address offset: 0x020

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be read with secure or nonsecure access.

If TZEN = 1, this register can be written only with secure access. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0], and BHKLOCK can be written with nonsecure access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on TAMP_PRIVCFGR configuration (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 44.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read, and can only be written to 1 by software. It is cleared by hardware together with the backup registers, following a tamper detection event, or when the readout protection (RDP) is disabled.

0: Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the protection zone they belong to.

1: Backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read, nor in write (they are read as 0, and write is ignored).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup register write protection offset

The value of this bitfield must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC - 1, with BKPWSEC ≥ BKPRWSEC).

If TZEN = 1, these backup registers can be written only with secure access. They can be read with secure or nonsecure access. If TZEN = 0, the protection zone 2 can be read and written with nonsecure access.

If BKPWSEC = 0, or if BKPWSEC ≤ BKPRWSEC, there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC, from 0 to 32). They can be read or written with secure or nonsecure access. If BKPWSEC = 32, there is no protection zone 3 (refer to *Figure 548: Backup registers protection zones*).

Note: If BKPWPRIV in TAMP_PRIVCFGR is set, BKPWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection

0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.

1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup register read/write protection offset

The value of this bitfield must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC - 1).

If TZEN = 1, these backup registers can be read and written only with secure access.

If TZEN = 0: the protection zone 1 can be read and written with nonsecure access.

If BKPRWSEC = 0, there is no protection zone 1 (refer to *Figure 548: Backup registers protection zones*).

Note: If BKPRWPRIV in TAMP_PRIVCFGR is set, BKPRWSEC[7:0] can be written only in privileged mode.

44.6.6 TAMP privileged configuration register (TAMP_PRIVCFGR)

Address offset: 0x024

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be written only with privileged access.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the configuration (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privileged protection (excluding backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 44.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privileged protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privileged protection
0: Backup registers zone 1 can be written with privileged or unprivileged access.
1: Backup registers zone 1 can be written only with privileged access.

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privileged protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written with privileged or unprivileged access.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written with privileged access.

Bits 14:0 Reserved, must be kept at reset value.

44.6.7 TAMP interrupt enable register (TAMP_IER)

Address offset: 0x02C

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.
			rw	rw	rw		rw	rw	rw	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
											rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ITAMPiIE**: Internal tamper i interrupt enable (i = 13 to 11)
0: Internal tamper i interrupt disabled
1: Internal tamper i interrupt enabled

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **ITAMPiIE**: Internal tamper i interrupt enable (i = 9 to 5)
0: Internal tamper i interrupt disabled
1: Internal tamper i interrupt enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled
1: Internal tamper 3 interrupt enabled

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiIE**: Tamper i interrupt enable (i = 5 to 1)
0: Tamper i interrupt disabled
1: Tamper i interrupt enabled

44.6.8 TAMP status register (TAMP_SR)

Address offset: 0x030

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.
			r	r	r		r	r	r	r	r		r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
											r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ITAMPiF**: Internal tamper i detection flag (i = 13 to 11)

This flag is set by hardware when a tamper detection event is detected on the internal tamper i.

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **ITAMPiF**: Internal tamper i detection flag (i = 9 to 5)

This flag is set by hardware when a tamper detection event is detected on the internal tamper i.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3F**: Internal tamper 3 detection flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiF**: Tamper i detection flag (i = 5 to 1)

This flag is set by hardware when a tamper detection event is detected on TAMPi input.

44.6.9 TAMP nonsecure masked interrupt status register (TAMP_MISR)

Address offset: 0x034

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*). This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.
			r	r	r		r	r	r	r	r		r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
											r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ITAMPiMF**: Internal tamper i nonsecure interrupt masked flag (i = 13 to 11)
This flag is set by hardware when the internal tamper i nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **ITAMPiMF**: Internal tamper i nonsecure interrupt masked flag (i = 9 to 5)
This flag is set by hardware when the internal tamper i nonsecure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiMF**: Tamper i nonsecure interrupt masked flag (i = 5 to 1)
This flag is set by hardware when the tamper i nonsecure interrupt is raised.

44.6.10 TAMP secure masked interrupt status register (TAMP_SMISR)

Address offset: 0x038

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.
			r	r	r		r	r	r	r	r		r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
											r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ITAMPiMF**: Internal tamper i secure interrupt masked flag (i = 13 to 11)
This flag is set by hardware when the internal tamper i secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **ITAMPiMF**: Internal tamper i secure interrupt masked flag (i = 9 to 5)
This flag is set by hardware when the internal tamper i secure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **TAMPiMF**: Tamper i secure interrupt masked flag (i = 5 to 1)
This flag is set by hardware when the tamper i secure interrupt is raised.

44.6.11 TAMP status clear register (TAMP_SCR)

Address offset: 0x03C

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.
			w	w	w		w	w	w	w	w		w		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
											w	w	w	w	w

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **CITAMPiF**: Internal tamper i detection flag clear (i = 13 to 11)
Writing 1 to this bit clears the ITAMPiF bit in TAMP_SR.

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **CITAMPiF**: Internal tamper i detection flag clear (i = 9 to 5)
Writing 1 to this bit clears the ITAMPiF bit in TAMP_SR.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CITAMP3F**: Internal tamper 3 detection flag clear
Writing 1 to this bit clears the ITAMP3F bit in TAMP_SR.

Bits 17:5 Reserved, must be kept at reset value.

Bits 4:0 **CTAMPiF**: Tamper i detection flag clear (i = 5 to 1)
Writing 1 to this bit clears the TAMPiF bit in TAMP_SR.

44.6.12 TAMP monotonic counter 1 register (TAMP_COUNT1R)

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]**: Monotonic counter

This register is read-only, and is incremented by one when a write access is done to it. This register cannot rollover, and is frozen when reaching the maximum value.

44.6.13 TAMP resource protection register (TAMP_RPCFGR)

Address offset: 0x054

Backup domain reset value: 0x8000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPCFG 31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG 5	RPCFG 4	RPCFG 3	RPCFG 2	RPCFG 1	Res.
										rw	rw	rw	rw	rw	

Bit 31 **RPCFG31**: Backup register protection

0: Backup registers and BHKLOCK are not included in device secrets protected by the TAMP.

1: Backup registers and BHKLOCK are included in device secrets protected by the TAMP.

Bits 30:6 Reserved, must be kept at reset value.

Bits 5:1 **RPCFGi**: Configurable resource i protection (i = 5 to 1)

0: Resource i is not included in device secrets protected by the TAMP.

1: Resource i is included in device secrets protected by the TAMP.

Note: Refer to tamp_confirmed_rpcfgi and tamp_potential_rpcfgi signals in Table 444: TAMP input/output pins and Table 446: TAMP interconnection.

Bit 0 Reserved, must be kept at reset value.

44.6.14 TAMP backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x4 * x (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset value: 0xUUUU UUUU

This register can be protected against nonsecure access (refer to *Section 44.3.5: TAMP secure protection modes*). This register can be protected against unprivileged access (refer to *Section 44.3.7: TAMP privilege protection modes*).

This register is not affected by System reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BKP[31:0]**: Backup data

The application can write or read data to and from these TAMP_BKPxR registers.

This register is protected with tamper detection when RPCFG31 is set in TAMP_RPCFGR:

– This register is reset on a confirmed tamper detection event, and is forced to reset value as long as there is at least one internal or external tamper flag being set.

– This register read and write access is blocked on a potential tamper detection event as long as there is at least one internal or external tamper flag being set.

This register is also reset when the readout protection (RDP) is disabled.

44.6.15 TAMP register map

Table 453. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TAMP_CR1**	Res.	Res.	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x004	TAMP_CR2	Res.	Res.	Res.	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	ITAMP3MSK	ITAMP2MSK	ITAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value				0	0	0	0	0	0	0				0	0	0												0	0	0	0	0

Table 453. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	**TAMP_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP3POM	ITAMP2POM	ITAMP1POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	Res.	ITAMP3POM	Res.	Res.
	Reset value																				0	0	0		0	0	0	0	0		0		
0x00C	TAMP_FLTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH [1:0]		TAMPPFLT [1:0]		TAMPFREQ [2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x020	**TAMP_SECCFGR**	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
	Reset value	0	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x024	**TAMP_PRIVCFGR**	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															
0x028	Reserved	Res.																															
0x02C	**TAMP_IER**	Res.	Res.	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x030	**TAMP_SR**	Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	Res.	ITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x034	**TAMP_MISR**	Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x038	**TAMP_SMISR**	Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x03C	**TAMP_SCR**	Res.	Res.	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	Res.	CITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value				0	0	0		0	0	0	0	0		0														0	0	0	0	0
0x040	**TAMP_COUNT1R**	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044-0x050	Reserved	Res.																															
0x054	**TAMP_RPCFGR**	RPCFGR31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFGR5	RPCFGR4	RPCFGR3	RPCFGR2	RPCFGR1	Res.
	Reset value	0																									0	0	0	0	0	
0x058-0x0FC	Reserved	Reserved																															

Table 453. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100 + 0x4 * x (x=0 to 31)	**TAMP_COUNT1R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

45 Inter-integrated circuit interface (I2C)

45.1 Introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

45.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode
- Autonomous functionality in Stop mode

For information on I2C instantiation, refer to *Section 45.3: I2C implementation*.

45.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 454. I2C implementation

I2C features(1)	I2C1	I2C2	I2C3
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X	X
Autonomous mode	X	X	X
Wake-up from Stop mode	X(2)	X(2)	X(3)
SMBus/PMBus	X	X	X

1. X = supported.
2. Wake-up supported from Stop 0 and Stop 1 modes.
3. Wake-up supported from Stop 0, Stop 1, and Stop 2 modes.

45.4 I2C functional description

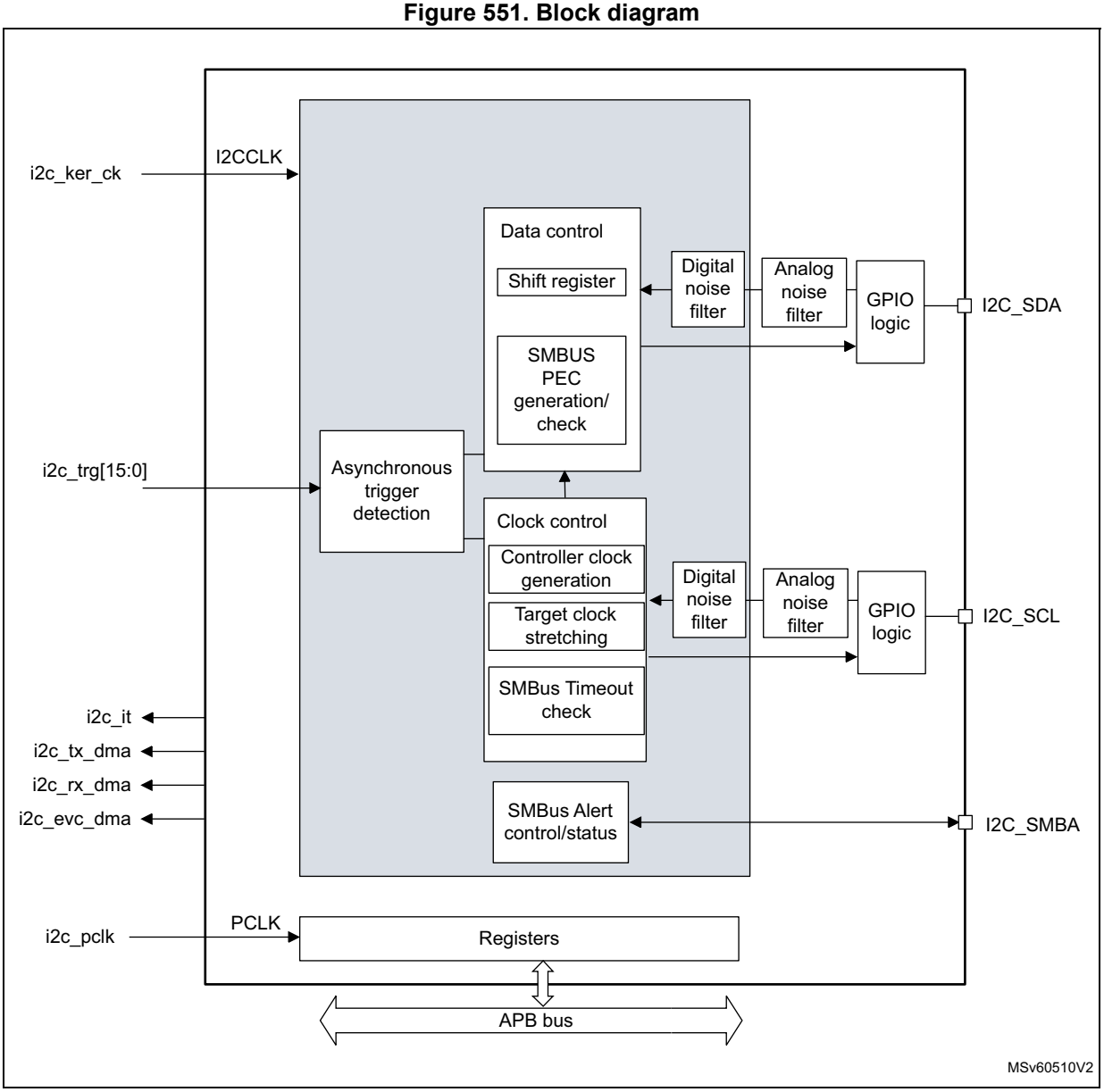
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

45.4.1 I2C block diagram

Figure 551. Block diagram



45.4.2 I2C pins and internal signals

Table 455. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 456. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_trg[15:0]	Input	I2C triggers
i2c_it	Output	I2C interrupts, refer to *Table 471* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)
i2c_evc_dma	Output	I2C event control DMA request (I2C_EVC)

Table 457. Trigger interconnections

Signal name	Source/destination	
	I2C1 and I2C2	I2C3
i2c_trg0	gpdma1_ch0_tc	gpdma1_ch0_tc
i2c_trg1	gpdma1_ch1_tc	gpdma1_ch1_tc
i2c_trg2	gpdma1_ch2_tc	gpdma1_ch2_tc
i2c_trg3	gpdma1_ch3_tc	gpdma1_ch3_tc
i2c_trg4	exti5	exti5
i2c_trg5	exti9	exti8
i2c_trg6	lptim1_ch1	lptim1_ch1
i2c_trg7	lptim2_ch1	lptim3_ch1
i2c_trg8	comp1_out	comp1_out
i2c_trg9	comp2_out	comp2_out
i2c_trg10	rtc_alra_trg	rtc_alra_trg
i2c_trg11	rtc_wut_trg	rtc_wut_trg
i2c_trg12	-	-
i2c_trg13	-	-
i2c_trg14	-	-
i2c_trg15	-	-

45.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

45.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

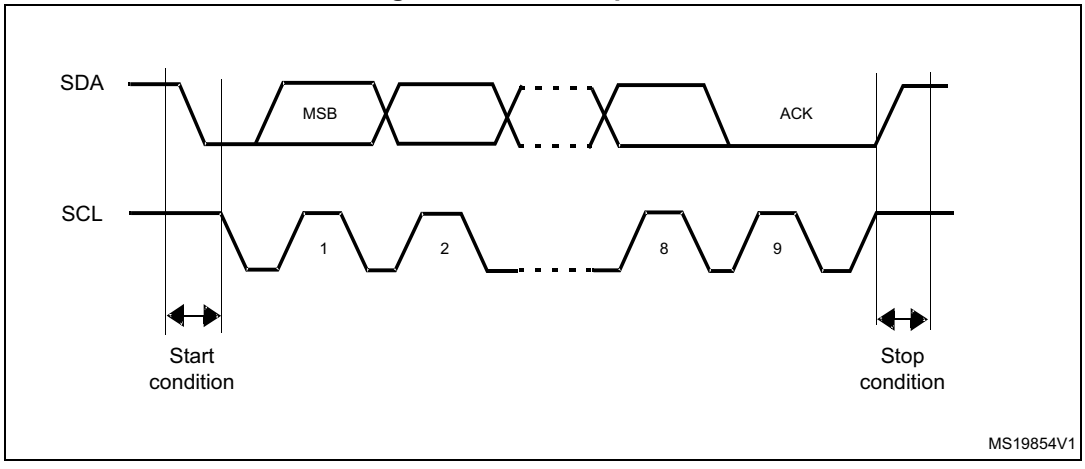
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 552. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

45.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 45.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 458. Comparison of analog and digital filters

Item	**Analog filter**	**Digital filter**
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Functionality in Stop mode not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

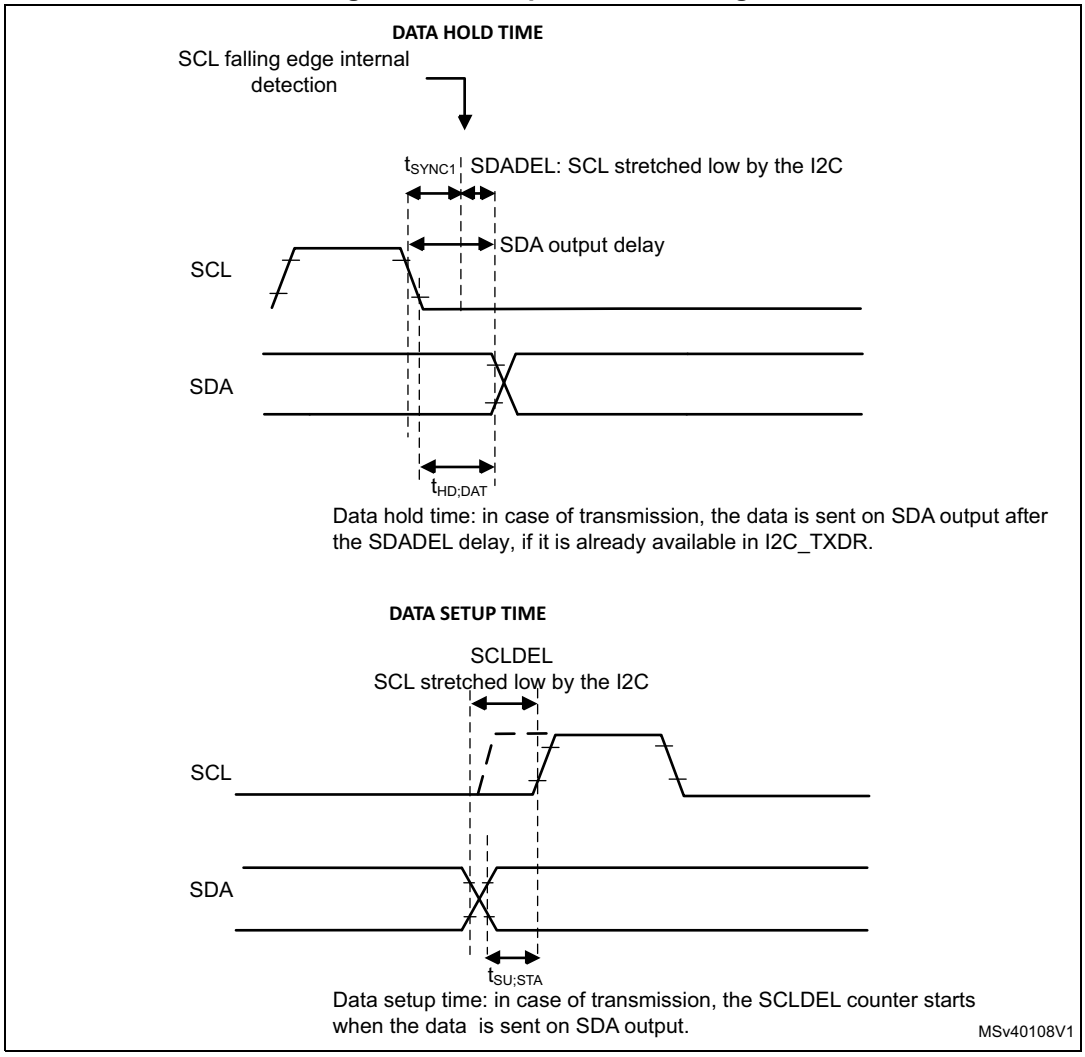
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 553. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\ (max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least $[(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCLK}$, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 459. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	

Table 459. I²C-bus and SMBus specification data setup and hold times (continued)

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

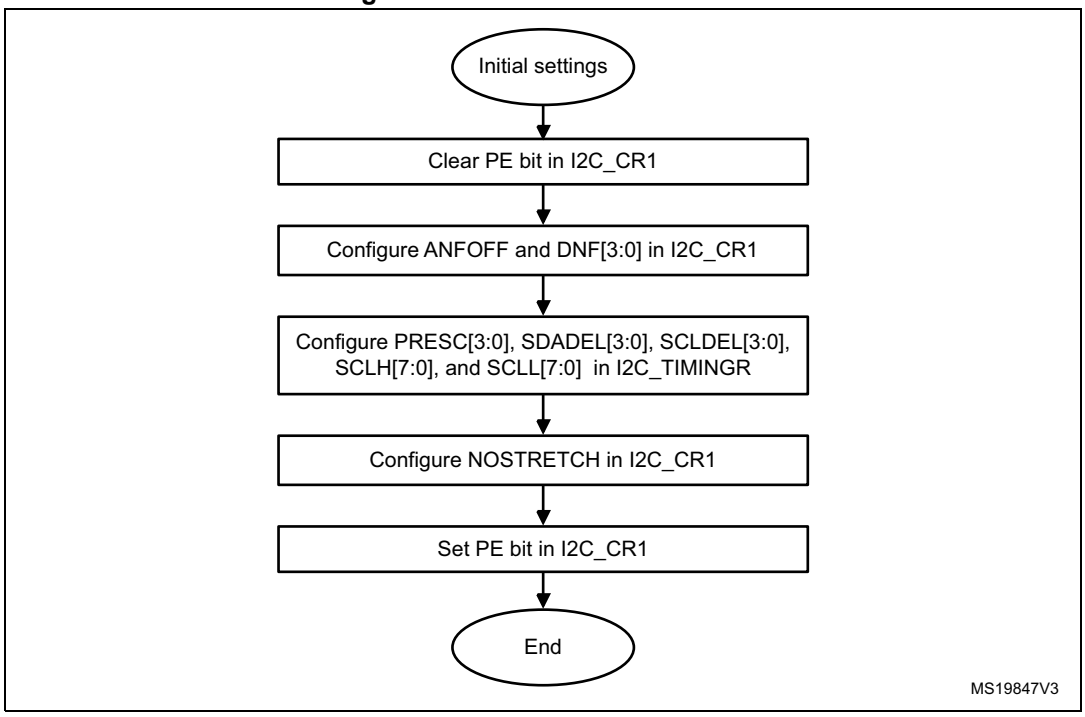
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 554. I2C initialization flow



45.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

45.4.7 I2C data transfer

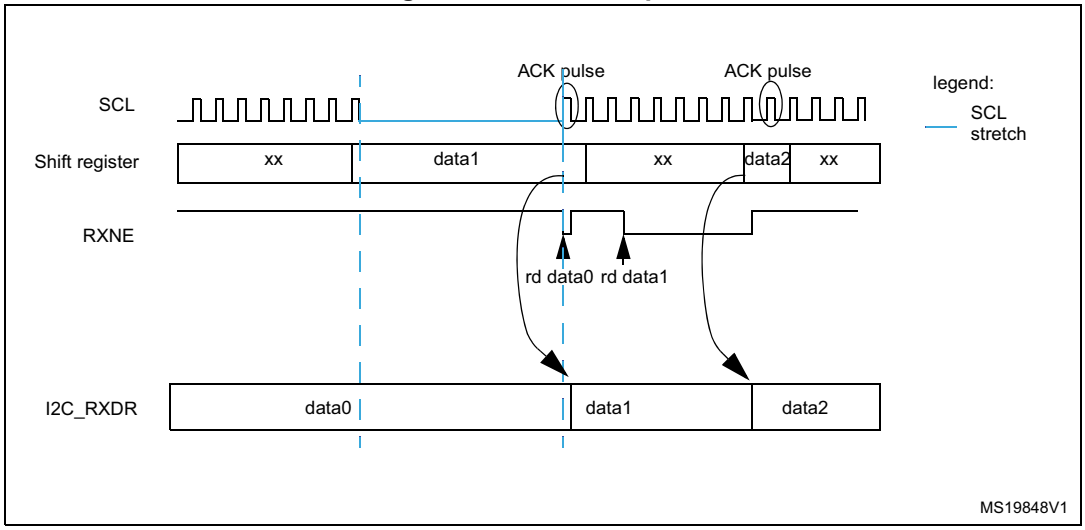
The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet

been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

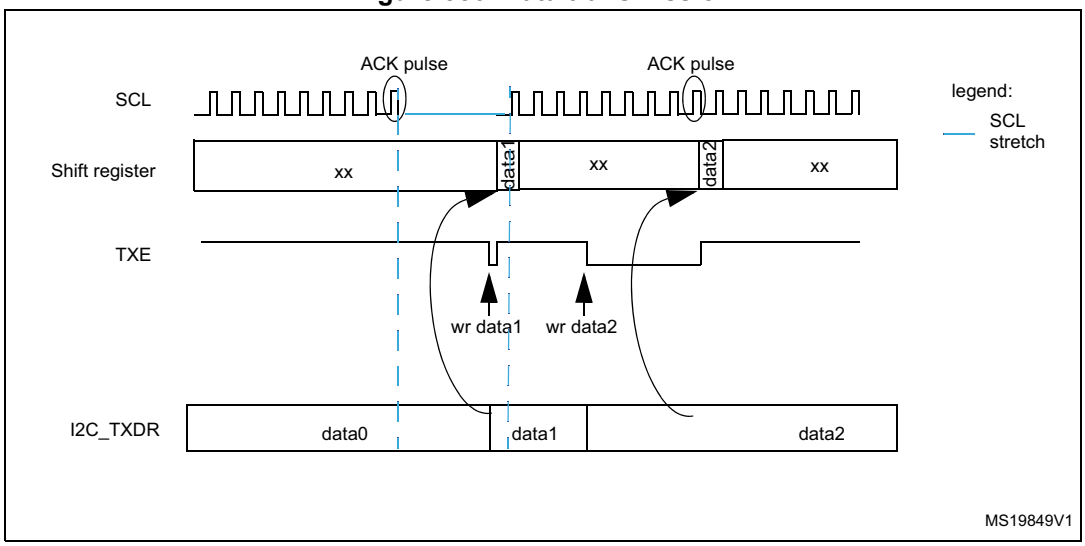
Figure 555. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 556. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 460. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

45.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

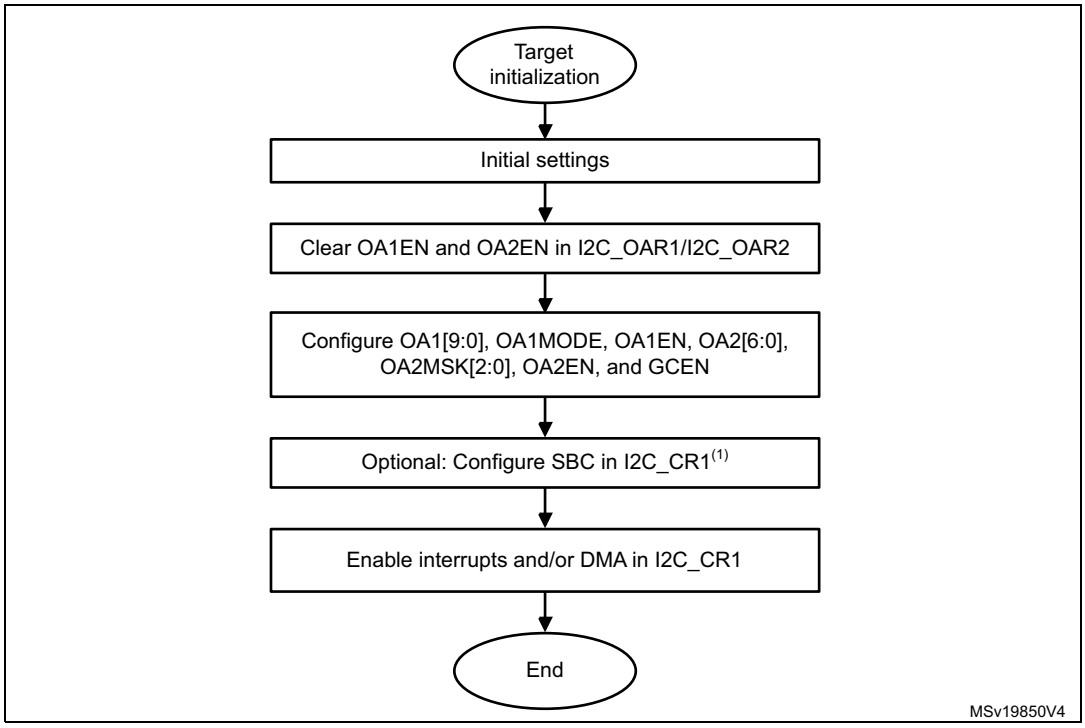
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 557. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the

first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 558. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

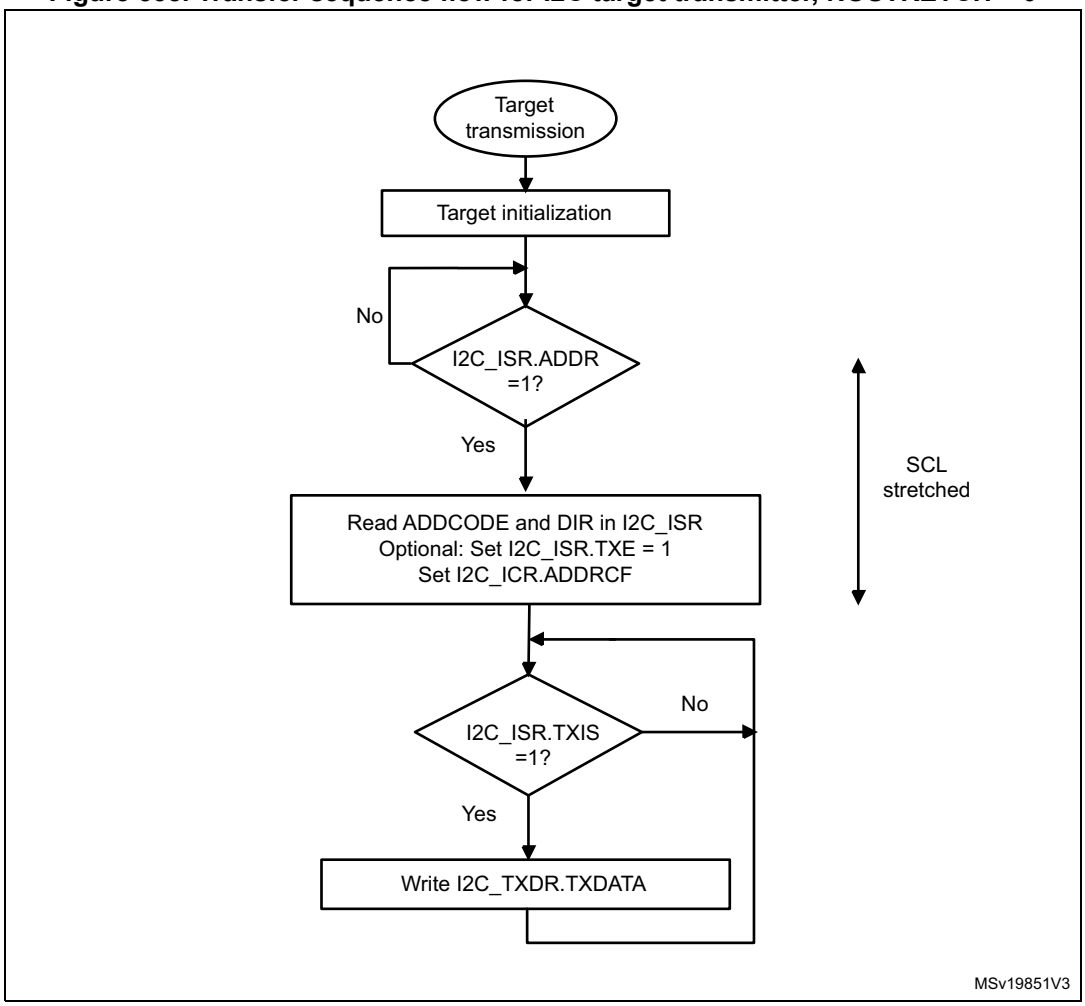


Figure 559. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

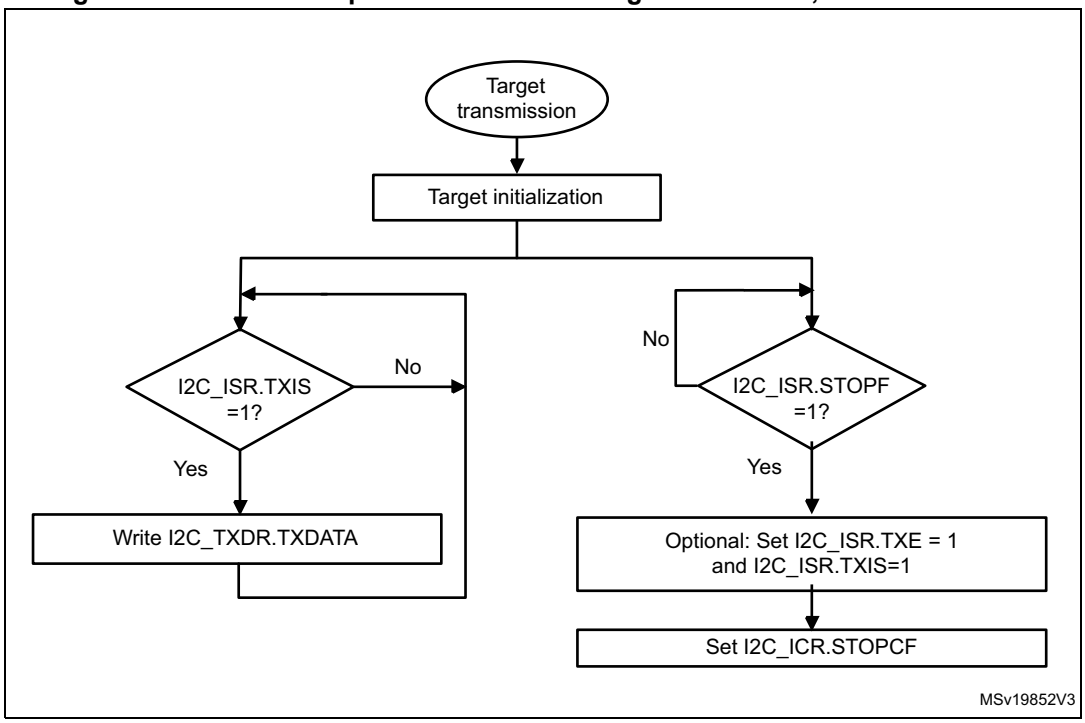
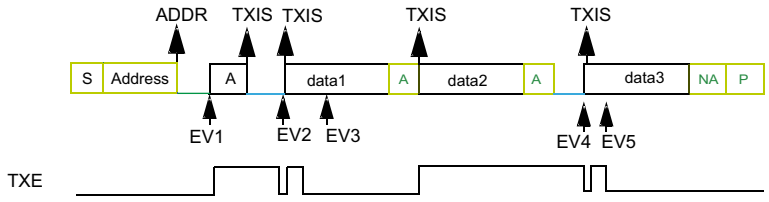


Figure 560. Transfer bus diagrams for I2C target transmitter (mandatory events only)

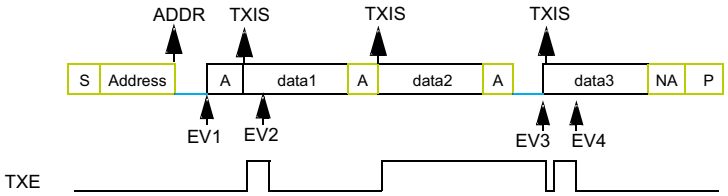
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

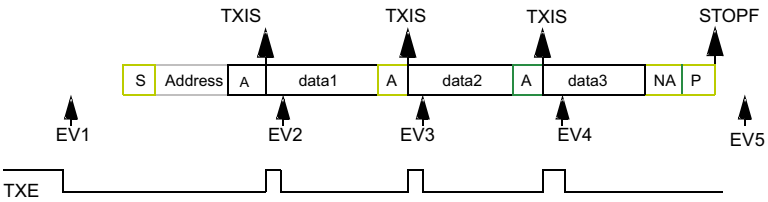
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 561. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

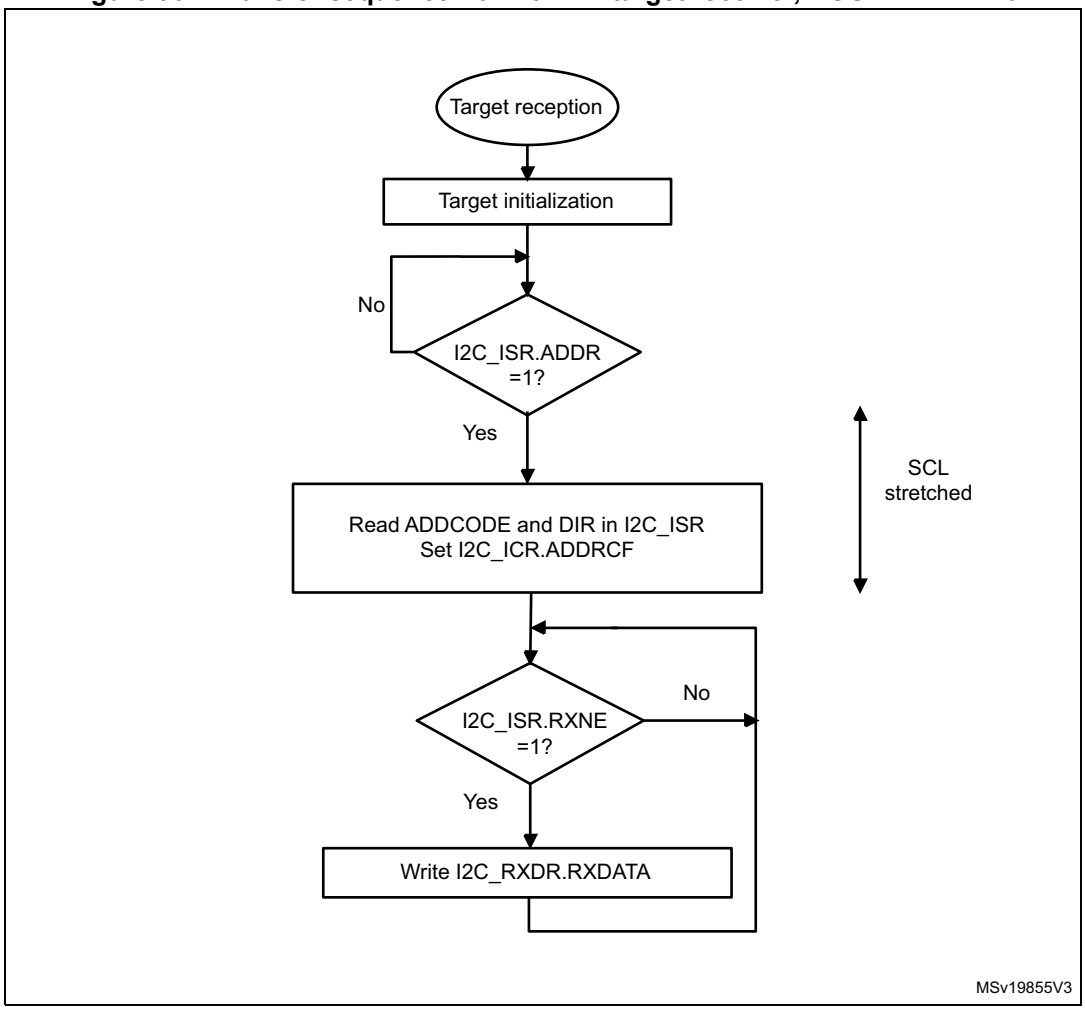


Figure 562. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

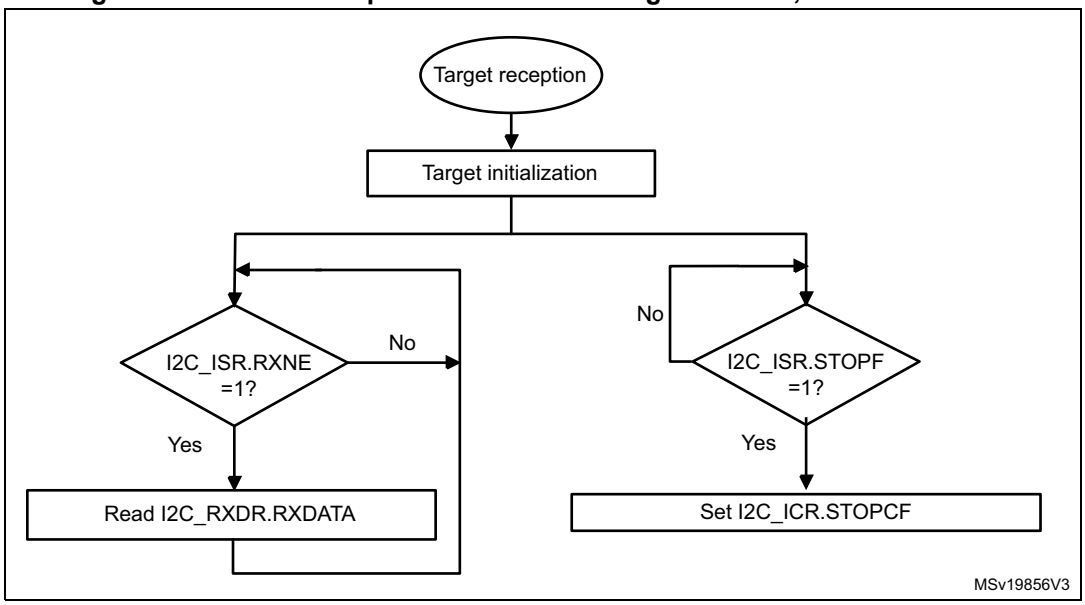
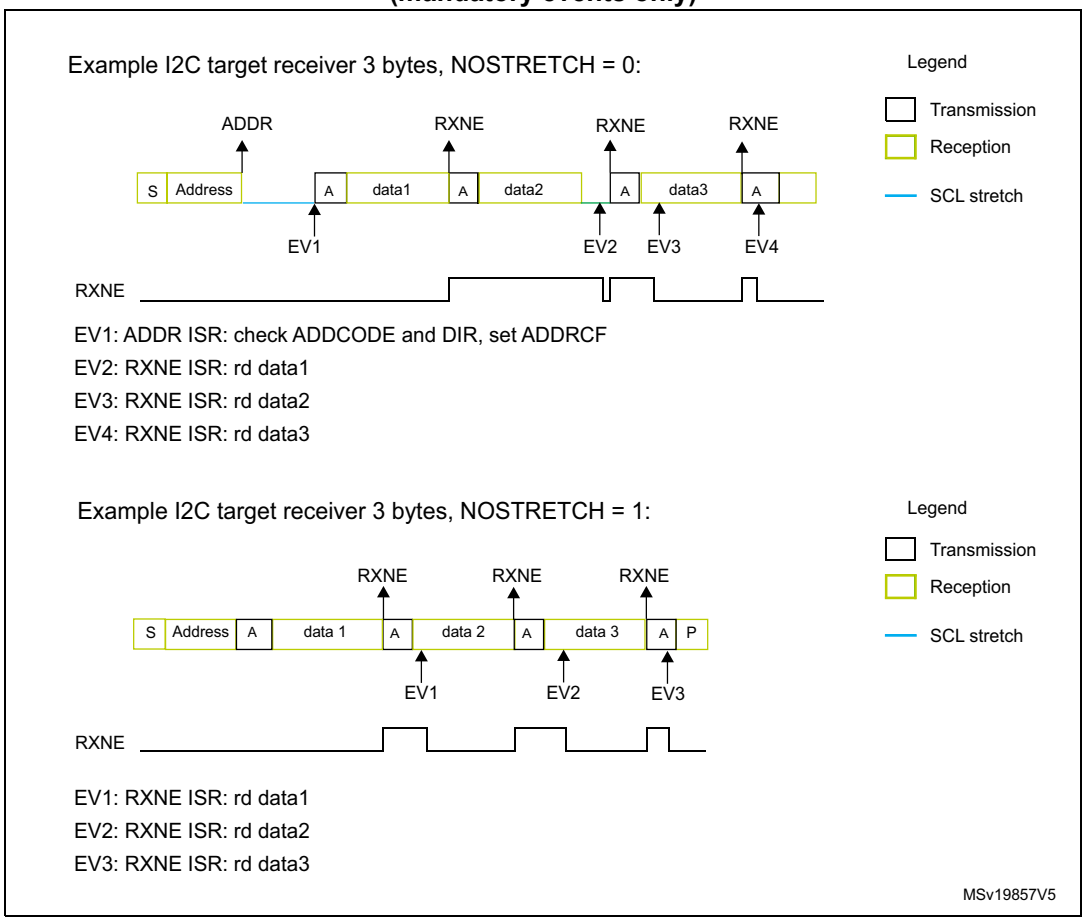


Figure 563. Transfer bus diagrams for I2C target receiver (mandatory events only)



45.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

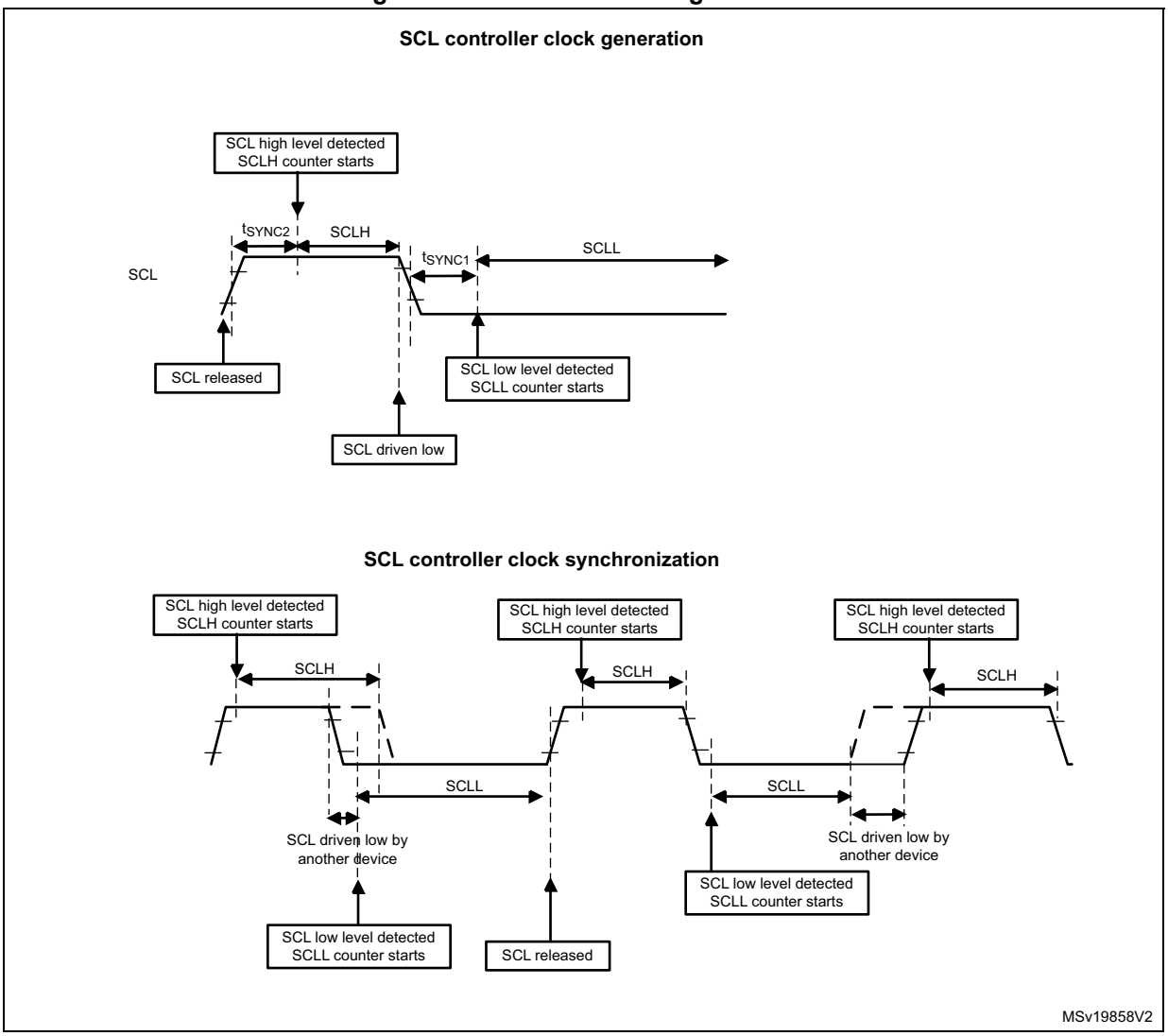
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 564. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 461. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 45.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

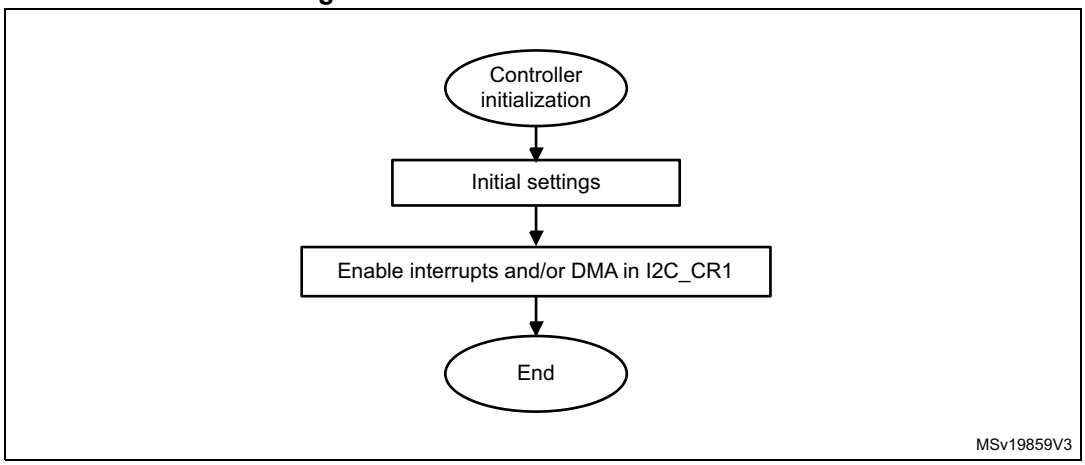
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 565. Controller initialization flow

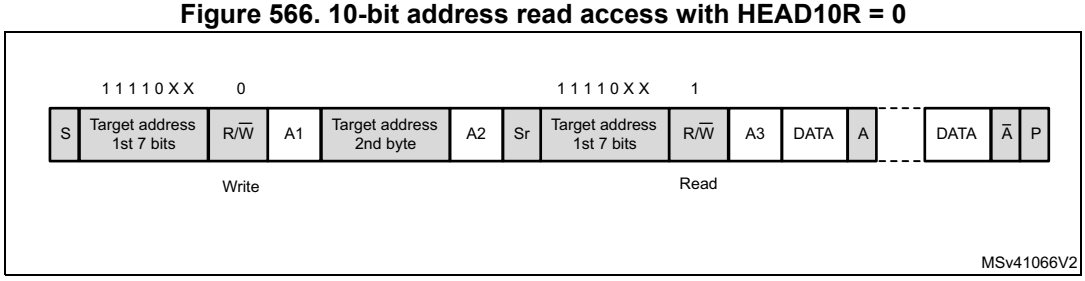


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

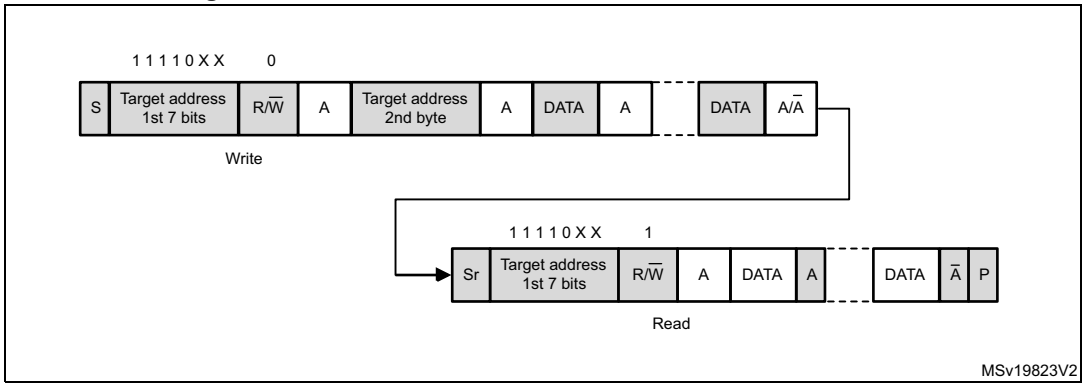
Figure 566. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 567. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 568. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

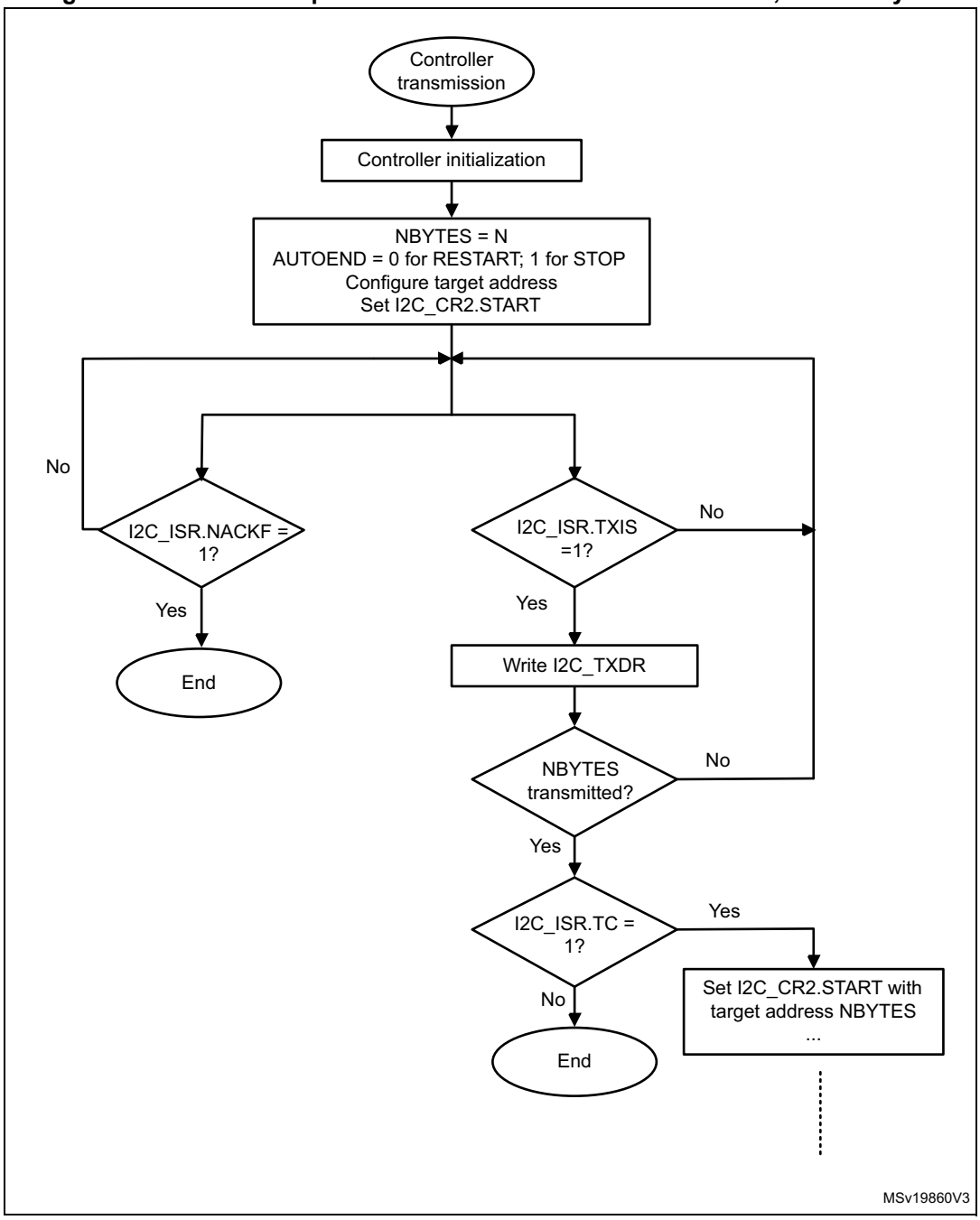


Figure 569. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

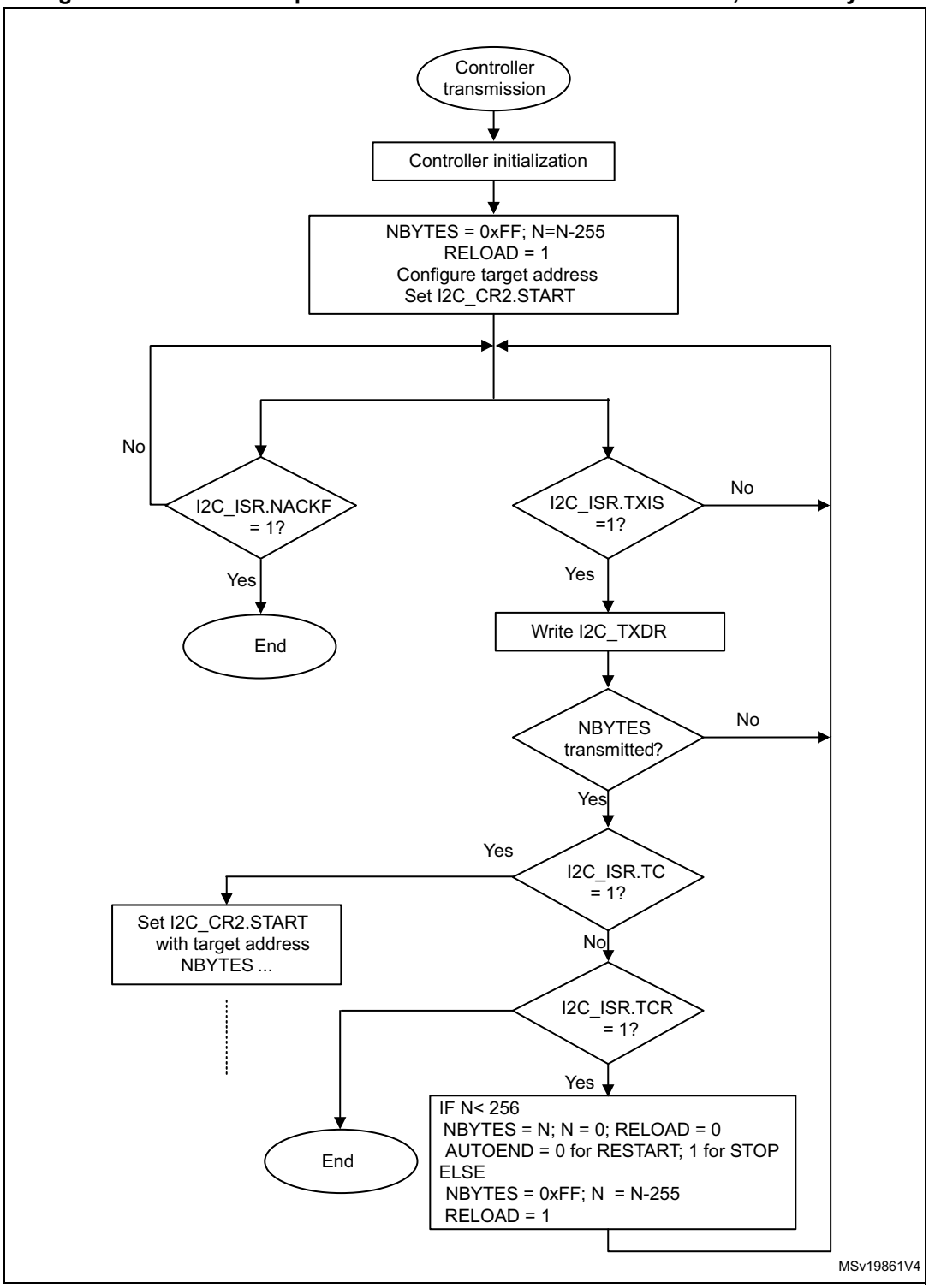
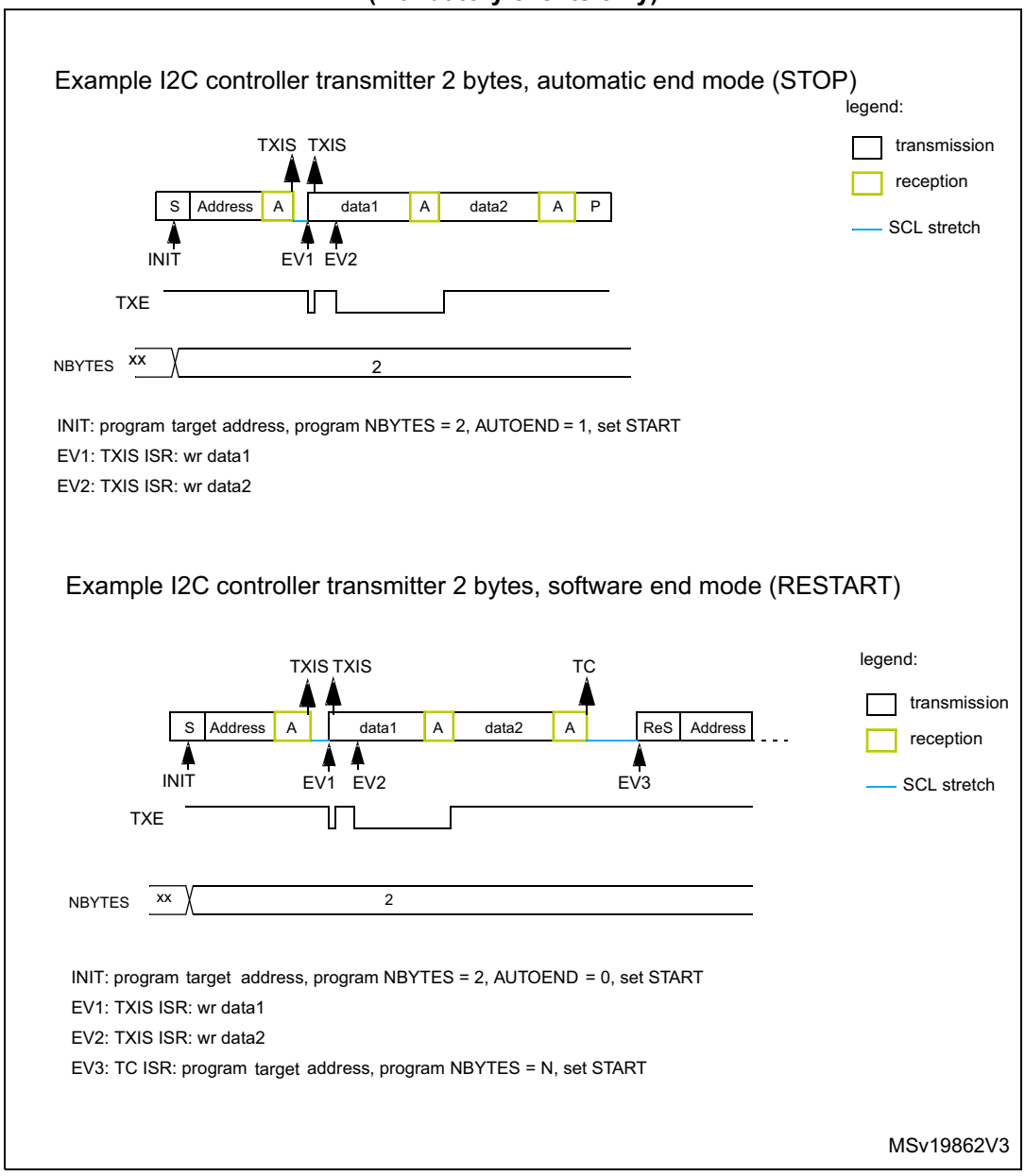


Figure 570. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 571. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

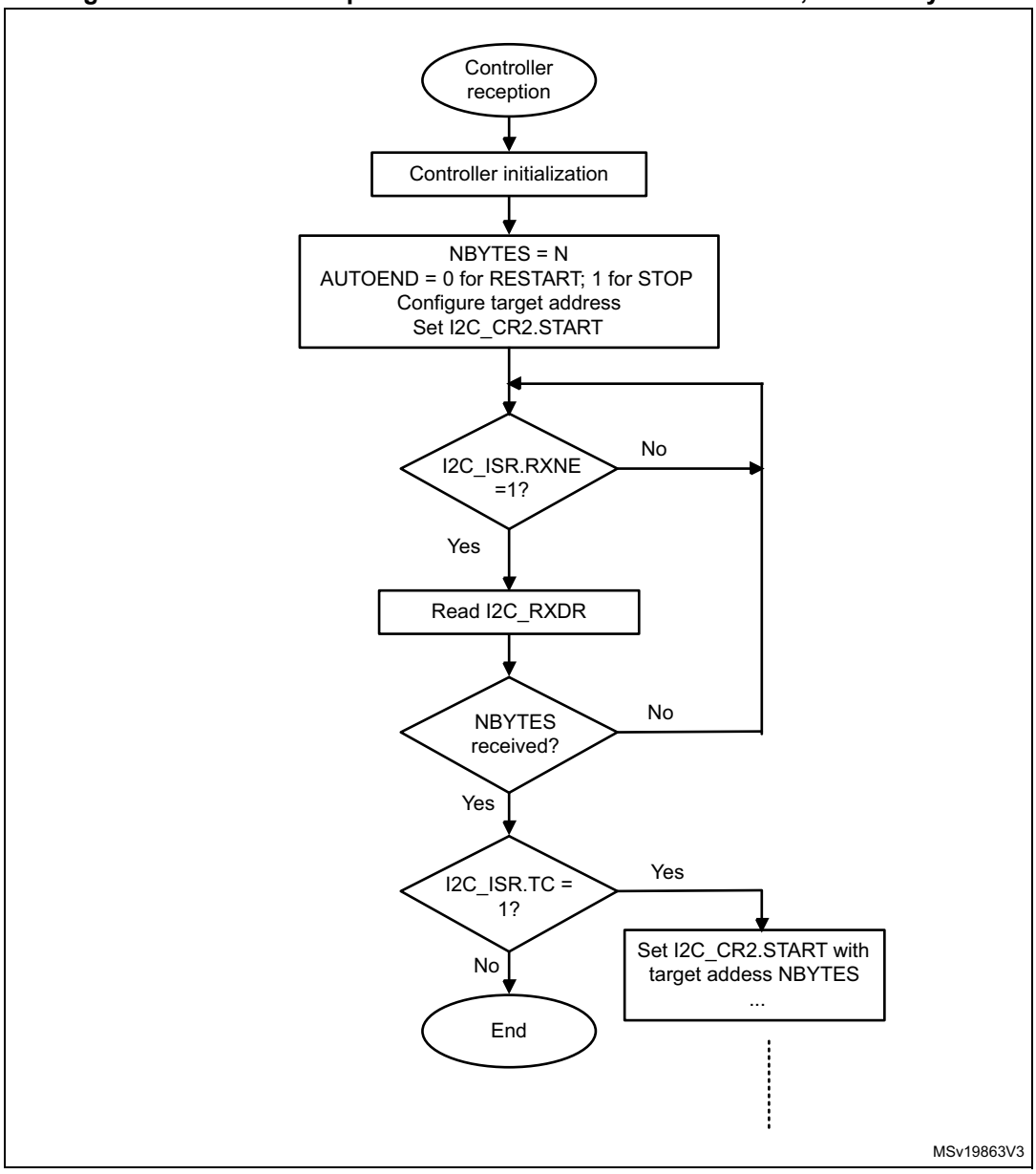


Figure 572. Transfer sequence flow for I2C controller receiver, N > 255 bytes

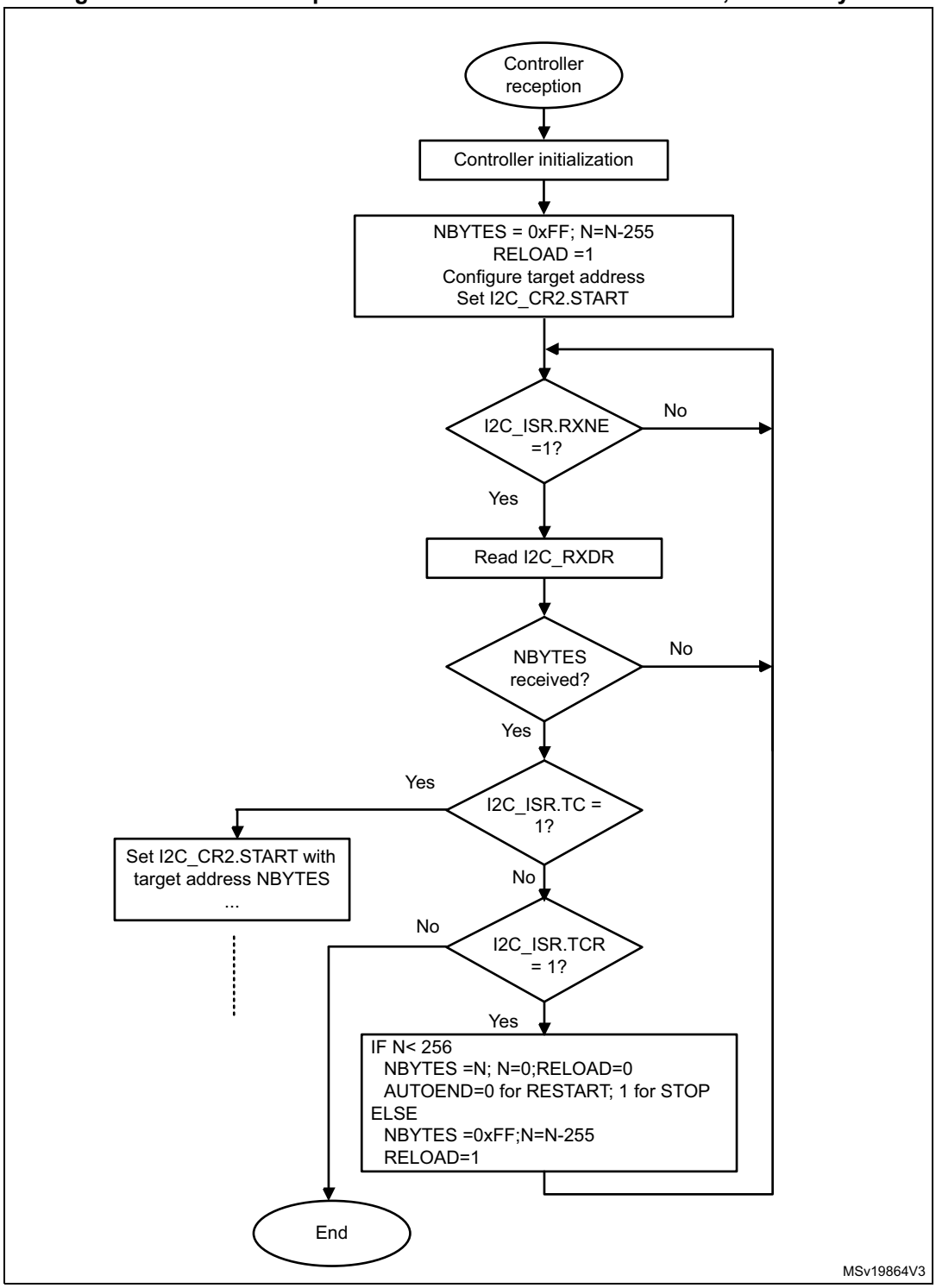
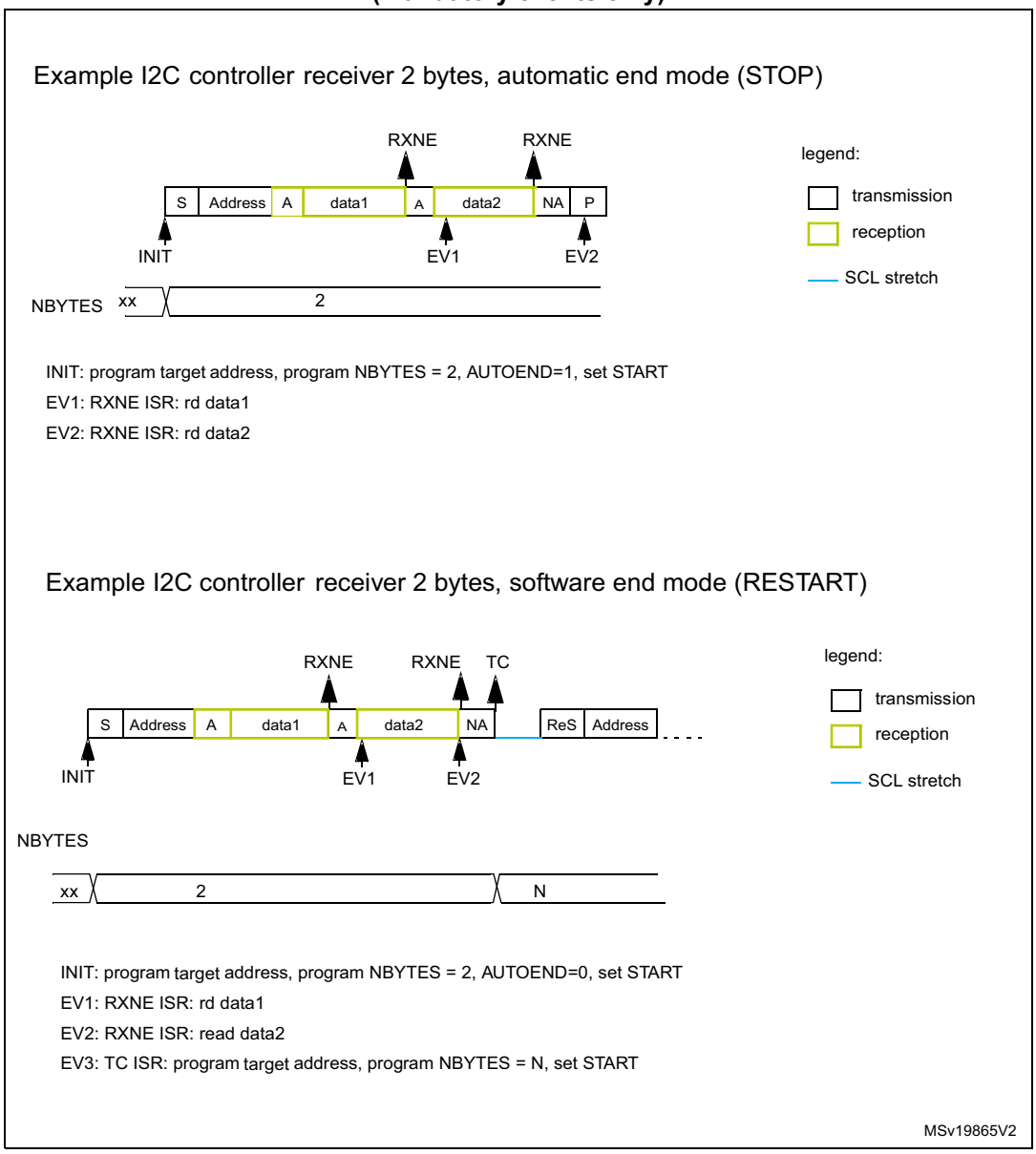


Figure 573. Transfer bus diagrams for I2C controller receiver (mandatory events only)



45.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 462. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 463. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

Table 464. Timing settings for f_{I2CCLK} of 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0xB	0xB	0x5	0x5
SCLL[7:0]	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH[7:0]	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~875 ns(4)
SDADEL[3:0]	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns
3. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns
4. t_{SYNC1} + t_{SYNC2} minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with t_{SYNC1} + t_{SYNC2} = 250 ns

45.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by

appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

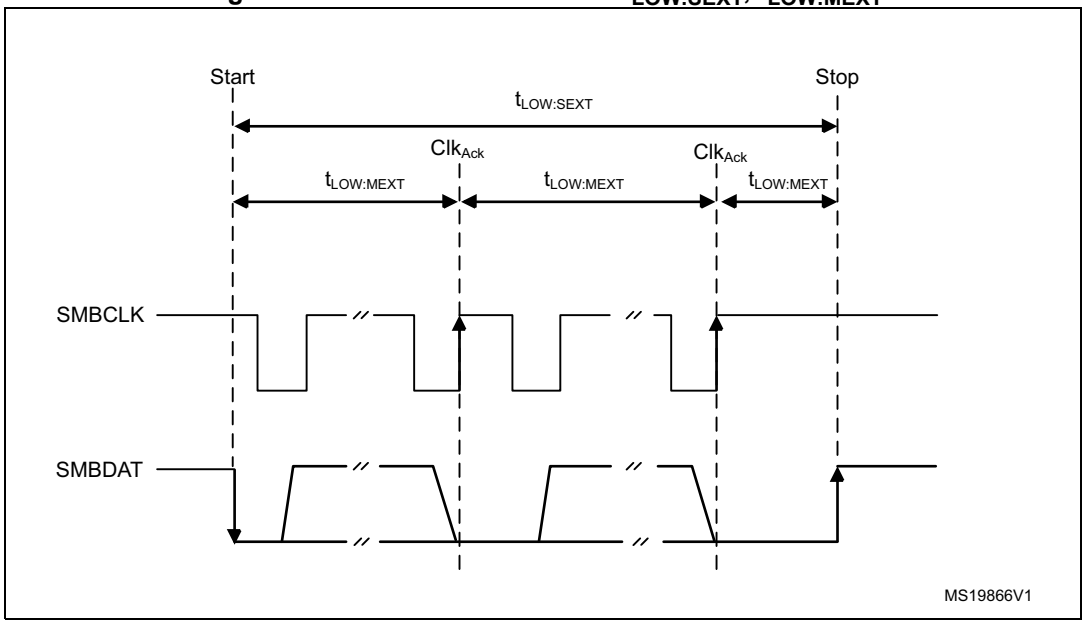
To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

Table 465. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.
2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 574. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 45.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

45.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 466. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 467*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 468*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 469*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

45.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 467. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

Table 468. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

Table 469. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

45.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 45.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 575. Transfer sequence flow for SMBus target transmitter N bytes + PEC

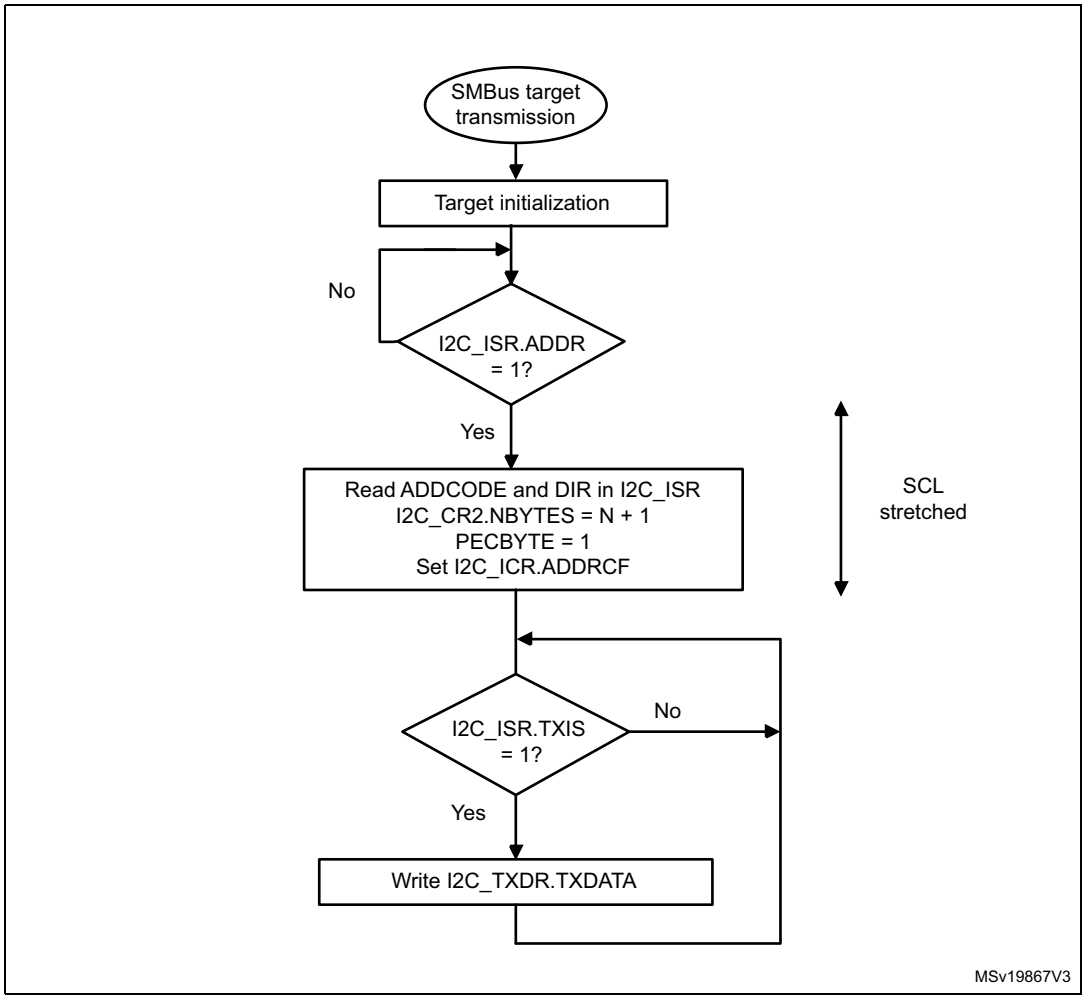
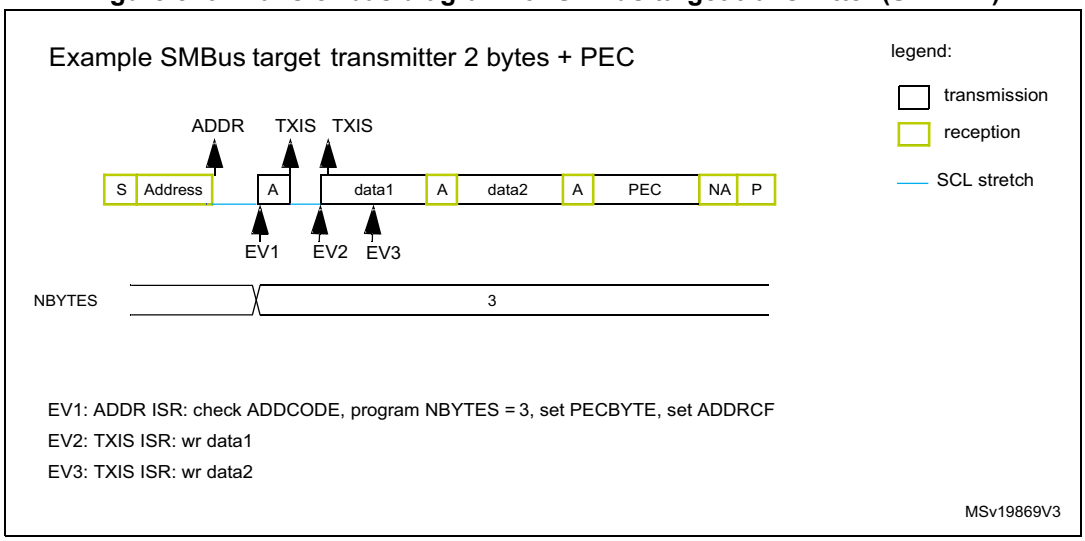


Figure 576. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 577. Transfer sequence flow for SMBus target receiver N bytes + PEC

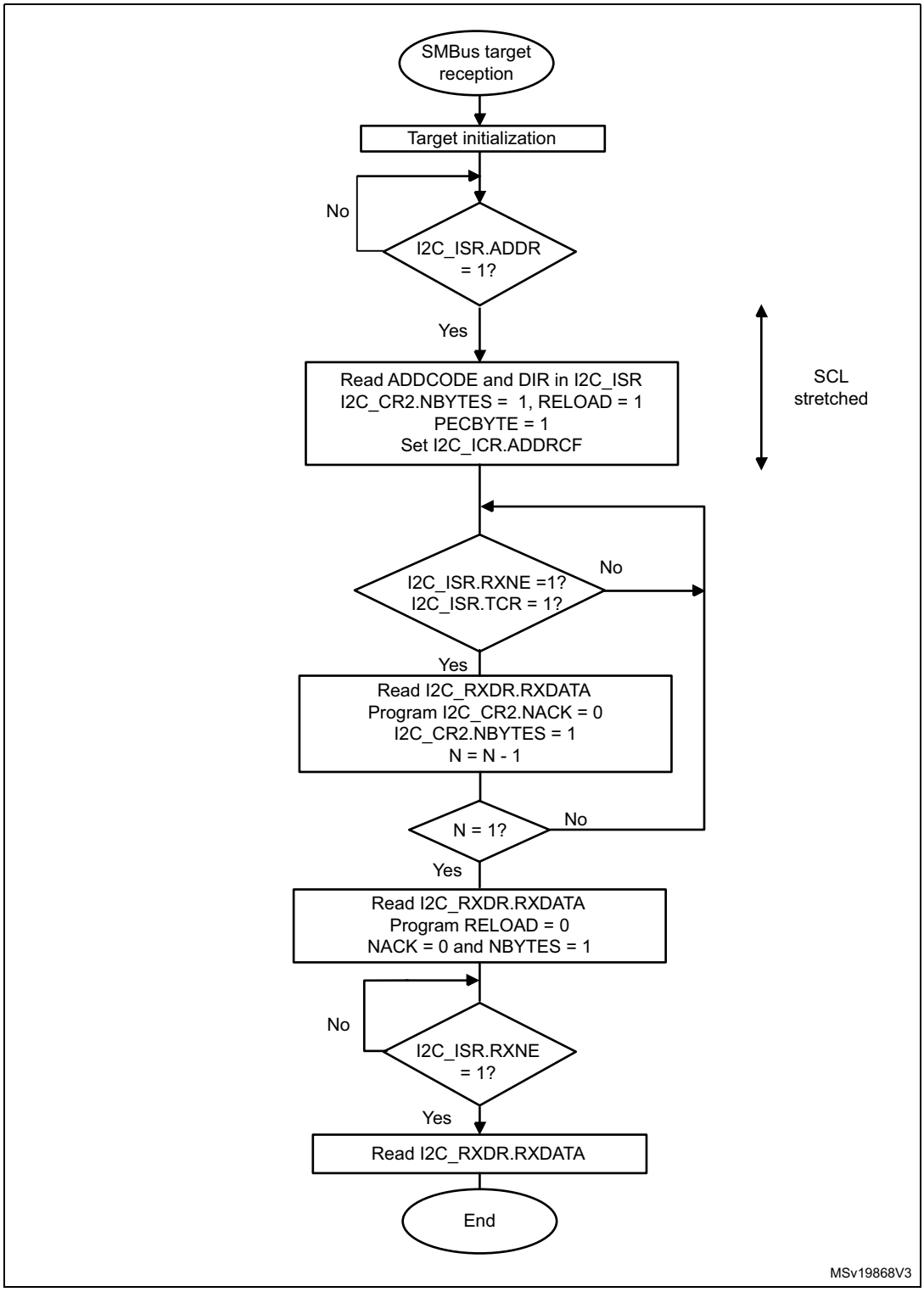
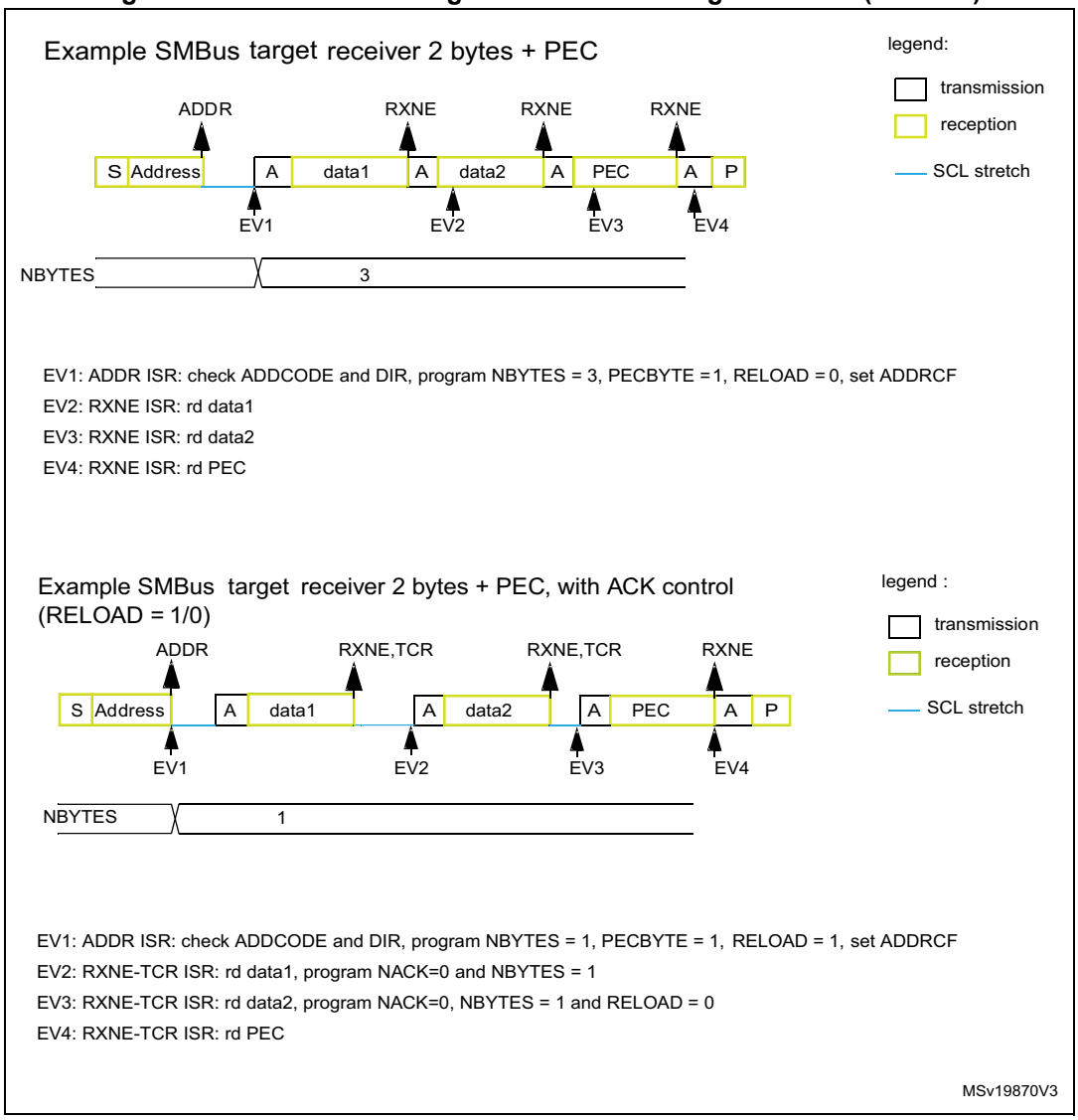


Figure 578. Bus transfer diagrams for SMBus target receiver (SBC = 1)



45.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 45.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

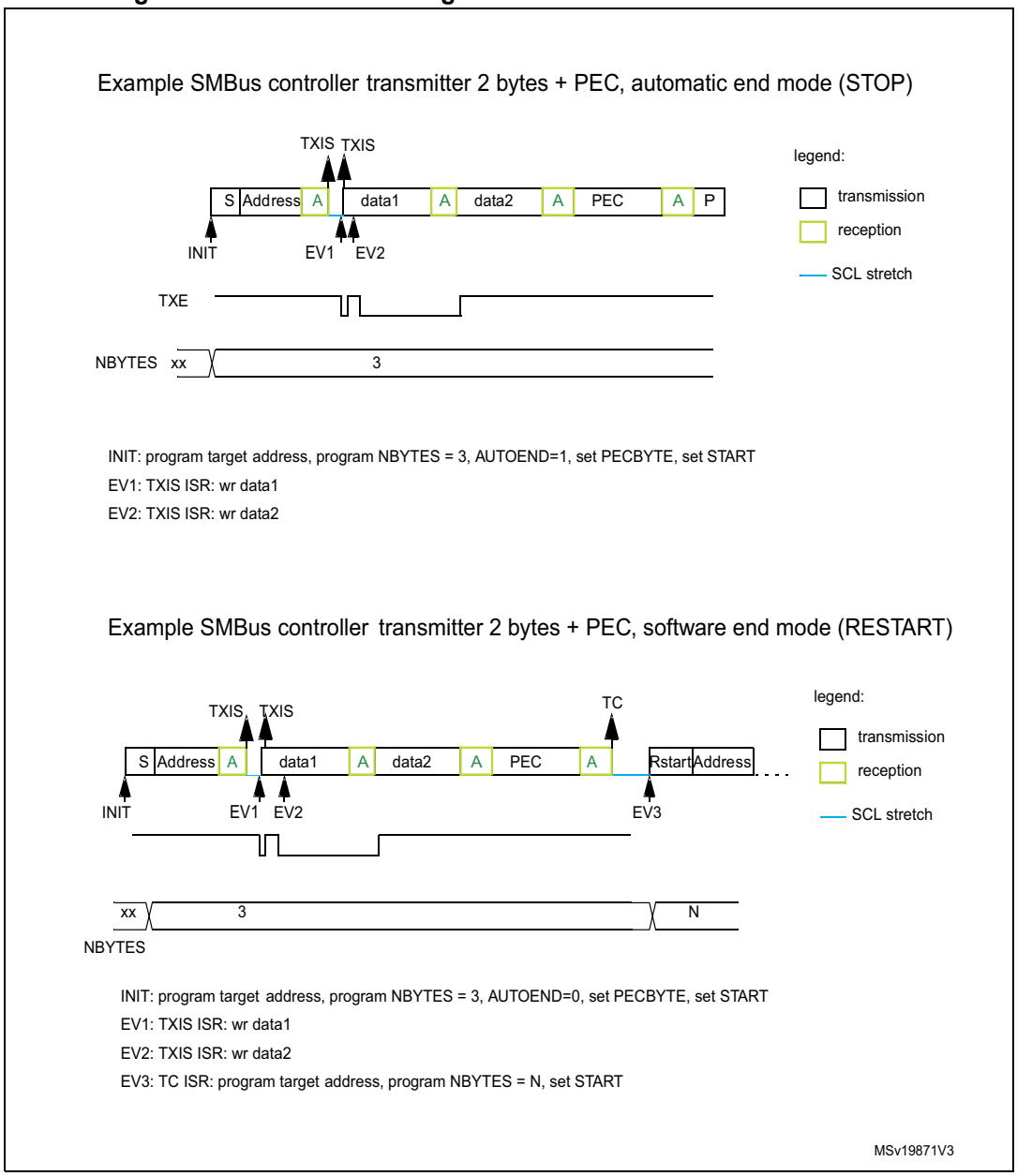
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 579. Bus transfer diagrams for SMBus controller transmitter



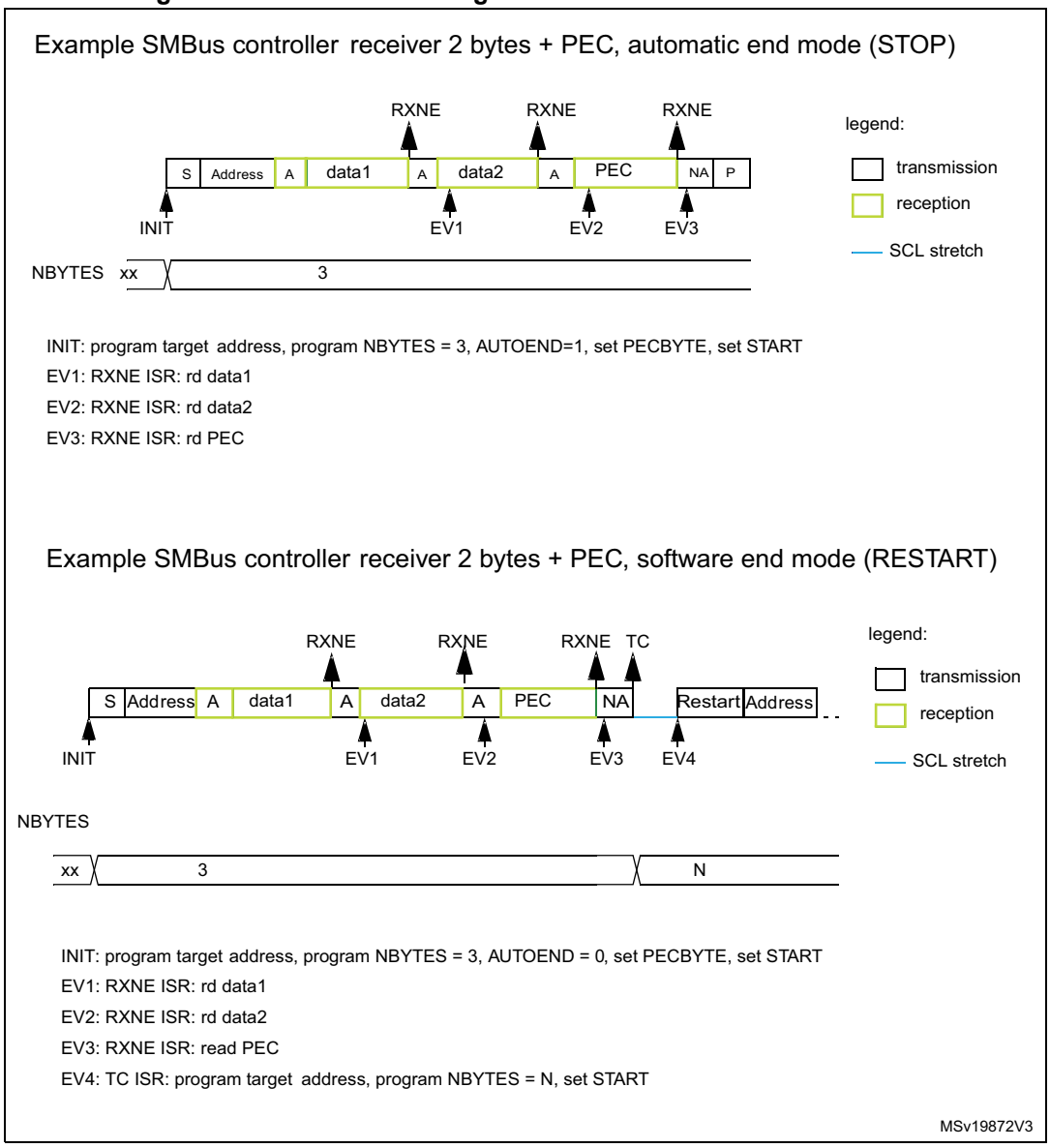
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 580. Bus transfer diagrams for SMBus controller receiver



45.4.16 Autonomous mode

The I2C peripheral can autonomously operate as controller or target in Stop mode as long as NOSTRETCH = 0. The autonomous mode is not supported when NOSTRETCH = 1. Besides Stop mode, the autonomous mode can also be used in Run and Sleep modes.

The APB clock is requested by the peripheral each time the I2C status needs to be updated. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the I2C configuration.

If an interrupt is generated, the device wakes up from Stop mode.

If there is no interrupt, the device remains in Stop mode, but the kernel and AHB/APB clocks are available for the I2C peripheral and all the autonomous peripherals enabled in the RCC.

If the DMA requests are enabled, the data are DMA-transferred to or from the SRAM, while the device remains in Stop mode.

Target mode

In target mode, the autonomous mode is enabled in Stop mode if WUPEN = 1 in the I2C_CR1. This mode is supported only when NOSTRETCH = 0.

The kernel clock is requested by the peripheral on START detection during all the transfer until the STOP condition occurs, when the target is addressed. If the target is not addressed, the kernel clock request is released after the address phase.

To optimize the functionality in target mode by using only DMA transfers, it is possible to set the ADDRACLR and STOPFACLR bits in the I2C_CR1 in order to avoid to serve the Address match (ADDR) and STOP detection (STOPF) events.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the internal oscillator is not switched on after a START is received in target mode.

Controller mode

In controller mode, a transfer can be automatically launched when an asynchronous trigger is detected in Run, Sleep or Stop mode. The trigger, selected by TRIGSEL in the I2C_AUTOCR register, generates a kernel clock request to allow the transfer, and launches a START condition and the I2C transfer as defined in the I2C_CR2 register. The kernel clock is requested until the STOP condition occurs.

To avoid waking up the CPU too often, it is possible to replace some interrupts by DMA requests, to make some control register write actions. In controller mode, *transfer complete* (TC) and *transfer complete reload* (TCR) events can generate a DMA request when the corresponding TCDMAEN or TCRDMAEN bits of the I2C_AUTOCR register are set. Consequently, the I2C_CR2 register can be written thanks to a DMA transfer, to program a new transfer (in TC event) or to reload the number of bytes (in TCR event).

In case a trigger is enabled, but the application needs to take the control back to start a different transfer, the following steps are required before writing in the I2C_CR2 register to launch the new transfer:

1. Wait for BUSY = 0 in the I2C_ISR register
2. Clear the TRIGEN bit of the I2C_AUTOCR register
3. Wait for longer than t_{BUF} (bus free time between a STOP and a START condition)
4. Wait for BUSY = 0 in the I2C_ISR register

Caution: When the device is in Stop mode, the I2C peripheral receives its kernel clock only when it is implicated in the transfer. Consequently, features such as timeouts and bus idle detection are not reliable in Stop mode.

Caution: The digital filter is not compatible with the functionality in Stop mode. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

45.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

45.5 I2C in low-power modes

Table 470. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. If the autonomous mode is enabled and I2C is clocked by an internal oscillator available in Stop mode: transfers in controller and in target modes are functional. DMA requests are functional, and the interrupts cause the device to exit the Stop mode. If WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 45.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific Stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

45.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 471. I2C interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Stop3, Standby, Shutdown mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	Yes(1)	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] = 0			
		Transfer complete	TC		Write START = 1 or STOP = 1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
	I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	Yes(1)	No
		Arbitration loss	ARLO		Write ARLOCF=1			
		Overrun/underrun	OVR		Write OVRCF=1			
		PEC error	PECERR		Write PECERRCF=1			
		Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF=1			
		SMBus alert	ALERT		Write ALERTCF=1			

1. Refer to the I2C implementation table for information about wake-up from Stop mode support per instance.

45.7 I2C DMA requests

45.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA

peripheral (see *Section 15: General purpose direct memory access controller (GPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

45.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 15: General purpose direct memory access controller (GPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

45.7.3 Controller event control using DMA

In controller mode, the transfer can be automatically managed while the device is in Run, Sleep or Stop mode, thanks to TC and TCR DMA requests.

If the TCDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when the TC bit of the I2C_ISR register is set. The DMA controller must be programmed to write the next command in the I2C_CR2 register. If both STOP and START bits are set in the new I2C_CR2 command, a STOP condition followed by a START condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2. If only START bit is set in the new I2C_CR2 command, a RESTART condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2.

If the TCRDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when TCR bit of the I2C_ISR register is set. The DMA controller must be programmed to write the remaining number of bytes to transfer in the I2C_CR2 register.

45.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

45.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

45.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus 20 mA drive enable
0: 20 mA I/O drive disabled
1: 20 mA I/O drive enabled

Bit 23 **PECEN:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable
0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).
Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disabled

1: Peripheral enabled

Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

45.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

45.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

45.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is

completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable

0: Own address 2 disabled. The received target address OA2 is NACKed.

1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

000: No mask

001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.

010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.

011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.

100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.

101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.

110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.

111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

45.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

45.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

45.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

45.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

45.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

45.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

45.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

45.9.12 I2C autonomous mode control register (I2C_AUTOCR)

Address offset: 0x2C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Trigger enable

0: Trigger disabled

1: Trigger enabled

When a trigger is detected, a START condition is sent and the transfer is launched as defined in I2C_CR2.

Bit 20 **TRIGPOL:** Trigger polarity

0: Trigger active on rising edge

1: Trigger active on falling edge

Note: This bit can be written only when PE = 0

Bits 19:16 **TRIGSEL[3:0]:** Trigger selection (refer to *Section 45.4.2: I2C pins and internal signals* I2C interconnections tables).
0000: i2c_trg0 selected
0001: i2c_trg1 selected
...
1111: i2c_trg15 selected
Note: This bit can be written only when PE = 0

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TCRDMAEN**: DMA request enable on Transfer Complete Reload event
0: DMA request not generated on Transfer Complete Reload event
1: DMA request generated on Transfer Complete Reload event

Bit 6 **TCDMAEN**: DMA request enable on Transfer Complete event
0: DMA request not generated on Transfer Complete event
1: DMA request generated on Transfer Complete event

Bits 5:0 Reserved, must be kept at reset value.

45.9.13 I2C register map

The table below provides the I2C register map and the reset values.

Table 472. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 472. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	I2C_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0									0	0						

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

46 Improved inter-integrated circuit (I3C)

46.1 Introduction

The I3C interface handles communication between this device and others, such as sensors and host processor, connected on an I3C bus.

An I3C bus is a two-wire, serial single-ended, multidrop bus, intended to improve a legacy I^2C bus.

The I3C SDR-only peripheral implements all the features required by the MIPI® I3C specification v1.1. It can control all I3C bus-specific sequencing, protocol, arbitration, and timing, and can act as controller (formerly known as master), or as target (formerly known as slave).

When acting as controller, the I3C peripheral improves the features of the I^2C interface preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C fast-mode (Fm) or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C peripheral can be used with DMA, to off-load the CPU.

46.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 479*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in *Table 485*)
  - I3C SDR read/write direct CCC messages (see details in *Table 485*)

- Frame-level management, when controller:
  - Software-triggered or hardware-triggered transfer
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 46.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 46.3.2*):
  - On a reset pattern
  - On a missed start
- Wake-up from Standby mode on a target reset pattern
- Autonomous mode and transfers during Stop mode(s) with DMA (see *Section 46.3.3*)
- Multiclock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 46.6.2*)

46.3 I3C implementation

46.3.1 I3C instantiation

There are several I3C instances in the device, all with the same features set:.

Table 473. I3C instantiation

Instance(s)
I3C1, I3C2

46.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 474*. For more details about the wake-up capabilities, refer to *Section 46.13*.

Table 474. I3C wake-up

Wake-up
From Standby mode
From Stop 0, Stop 1, Stop 2, and Stop 3 modes

46.3.3 I3C autonomous mode with DMA in Stop mode(s)

The I3C peripheral can be assisted with an autonomous DMA, functional in Stop mode, so that I3C transfer(s) is (are) performed over the I3C bus, as written to and/or read from the I3C FIFO-based registers via the programmed DMA channels, without involving the CPU. Afterwards, the I3C or/and DMA can be used to wake up the device from Stop mode.

Table 473 indicates the I3C instance(s) and DMA(s) able to operate autonomously in a Stop mode.

Table 475. I3C autonomous mode with DMA in Stop mode(s)

Instance	Autonomous mode with DMA
I3C1, I3C2	In Stop 0 and Stop 1 modes (with GPDMA1)

46.3.4 I3C FIFOs

The FIFOs are implemented as defined in *Table 476*.

Table 476. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

46.3.5 I3C triggers

During initialization (when EN = 0 in I3C_CFGR register), a hardware trigger signal can be configured (via TRIGHWEN, TRIGPOL, and TRIGSEL[3:0] of the I3C_CFGR register) so that the I3C peripheral, when acting as controller, can issue a frame transfer on the I3C bus once detected a trigger hit (rising/falling edge) of the enabled and selected hardware trigger.

The I3C trigger mapping is specified in *Table 477*.

Table 477. I3C triggers, when I3C as controller

TRIGSEL[0:3] in I3C_CFGR register	Selected I3C trigger
0	gpdma1_ch0_tc
1	gpdma1_ch1_tc
2	gpdma1_ch2_tc
3	gpdma1_ch3_tc
4	exti5
5	exti9
6	lptim1_ch1
7	lptim2_ch1
8	comp1_out
9	comp2_out
10	rtc_alara_trg
11	rtc_wut_trg
12	tim15_trgo
13	tim16_oc1
14	-
15	-

46.3.6 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 478*.

Table 478. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

46.3.7 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 479*.

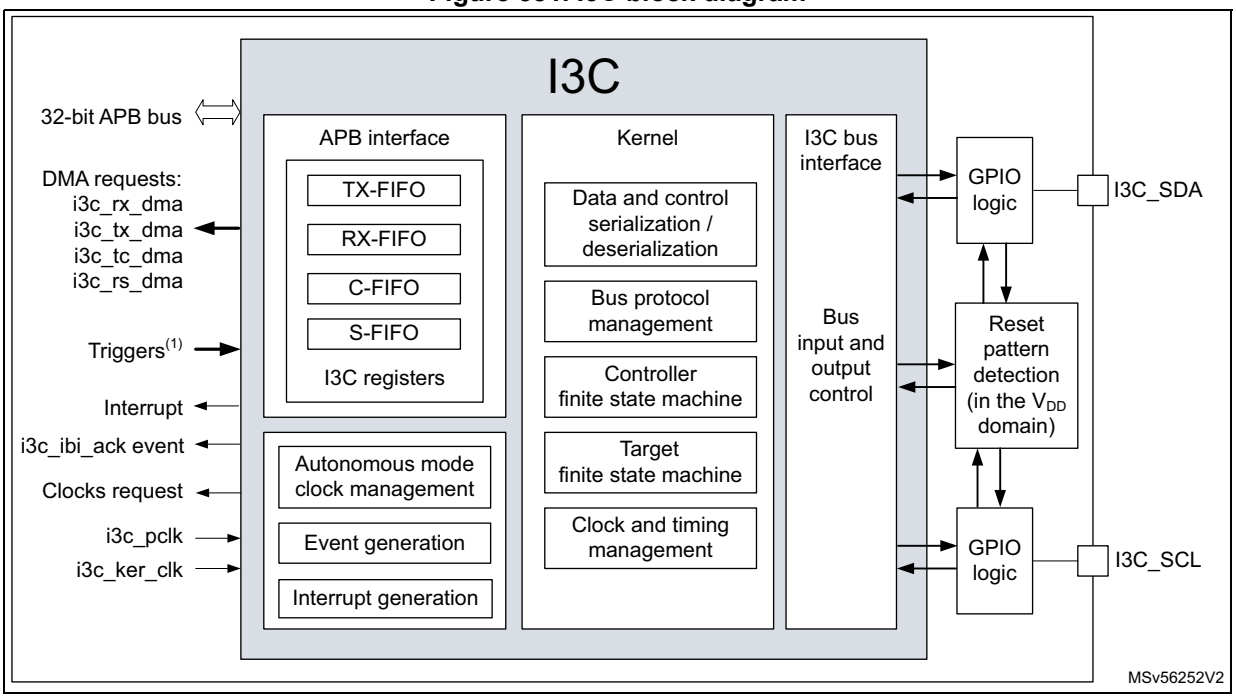
Table 479. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Fm/Fm+)	X	X	-	Mandatory when controller, and the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	-
Static address	X	X	-	No (intended) support of the I3C peripheral as a target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 485* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller (refer to *Timing control and sensor acquisition timestamping when I3C controller*). Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

46.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 581*.

Figure 581. I3C block diagram



1. Refer to *Section 46.3.5: I3C triggers*.

46.5 I3C pins and internal signals

Table 480. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	I3C bus serial data line
I3C_SCL	Input/output	I3C bus serial clock line

Table 481. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line

Table 481. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_trg[15:0]	I	Triggers, see *Table 477* for the mapping
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller
i3c_ibi_ack	O	IBI acknowledge event when the I3C peripheral acts as controller, for timing control with asynchronous mode 0

1. Refer to *Section 46.3.6: I3C interrupt(s)*.

46.6 I3C reset and clocks

46.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 485* and *Section 46.16.17* for more details.

This reset interrupt notification can be used to wake up from a low power mode, where the I3C peripheral (typically in the V_{CORE} domain) is active.

Finally, in an advanced low-power mode such as Standby, where the I3C peripheral is powered off but the device is in retention mode (V_{DD} is powered on), the I3C reset pattern detection block can be enabled via the PWR module, and used to wake up the device.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

46.6.2 I3C clocks and requirements

As indicated in *Figure 581*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE}

for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to *Section 10: Reset and clock control (RCC)*).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

46.7 I3C peripheral state and programming

46.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 582* and *Figure 583*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different configuration registers, and switches to the (enabled and) idle state.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

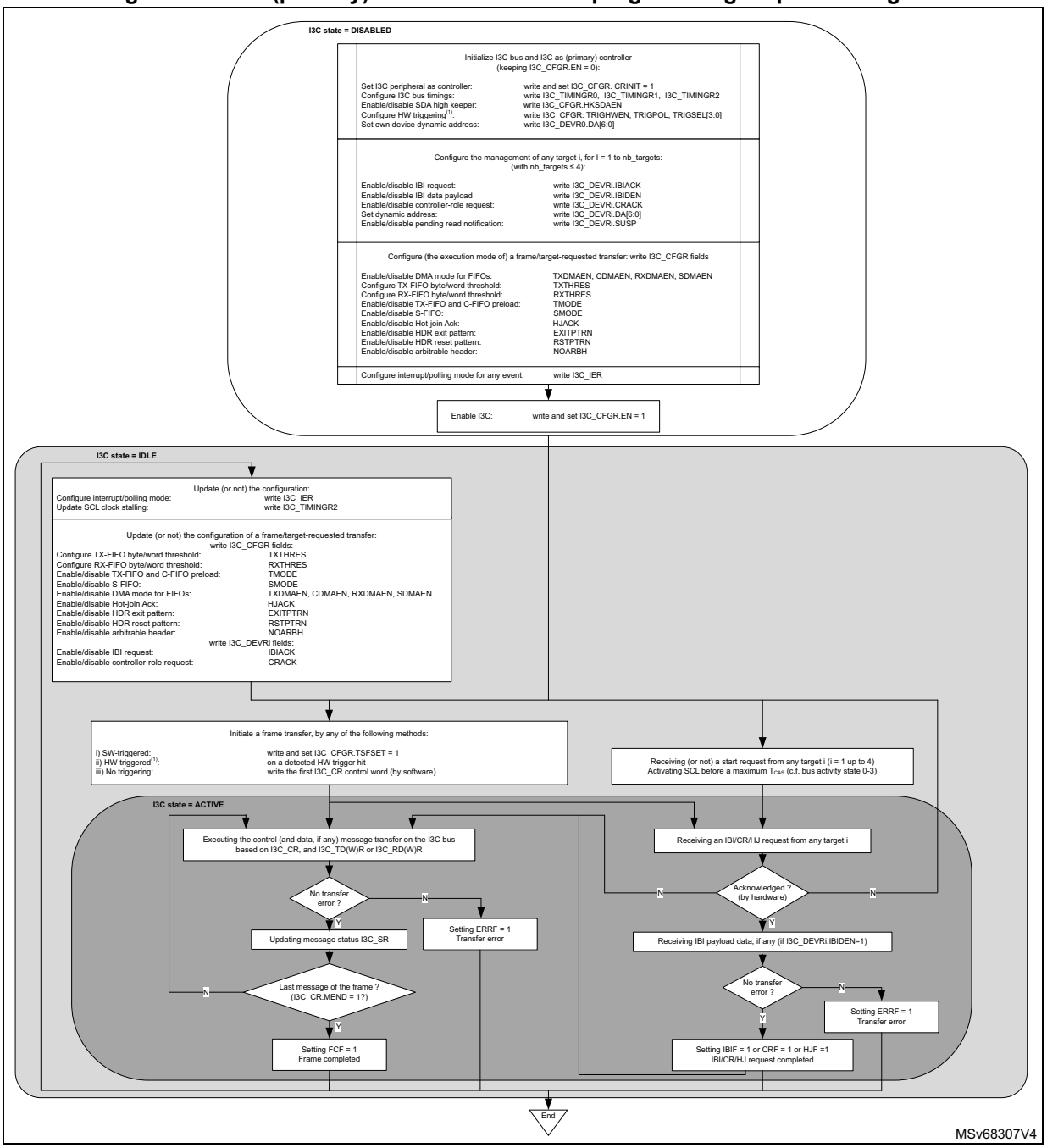
Note: The software can disable the I3C peripheral *(write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.*

Note: The software can disable the generation of the FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 46.3), to stream a series of frames.

46.7.2 I3C controller state and programming sequence

Figure 582 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 582. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 46.3.5: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
  - TRIGHWEN, TRIGPOL, TRIGSEL[3:0]: hardware-trigger mode
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
  - FCFDIS: FCF generation enable/disable

- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- Hardware-triggering (TRIGHWEN = 1 in the I3C_CFGR register): on a detected/hit trigger of the selected and enabled trigger signal
  - The trigger hit causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 46.9*:

- *Figure 594: IBI transfer, as controller/target*
- *Figure 595: Hot-join request transfer, as controller/target*
- *Figure 596: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR

register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 46.9*:

- *Figure 584: I3C CCC messages, as controller*
- *Figure 585: I3C broadcast ENTDAA CCC, as controller*
- *Figure 586: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 591: I3C private read/write messages, as controller*
- *Figure 593: Legacy I2C read/write messages, as controller*

Figure 582 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 46.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 46.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
  - FCFDIS

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

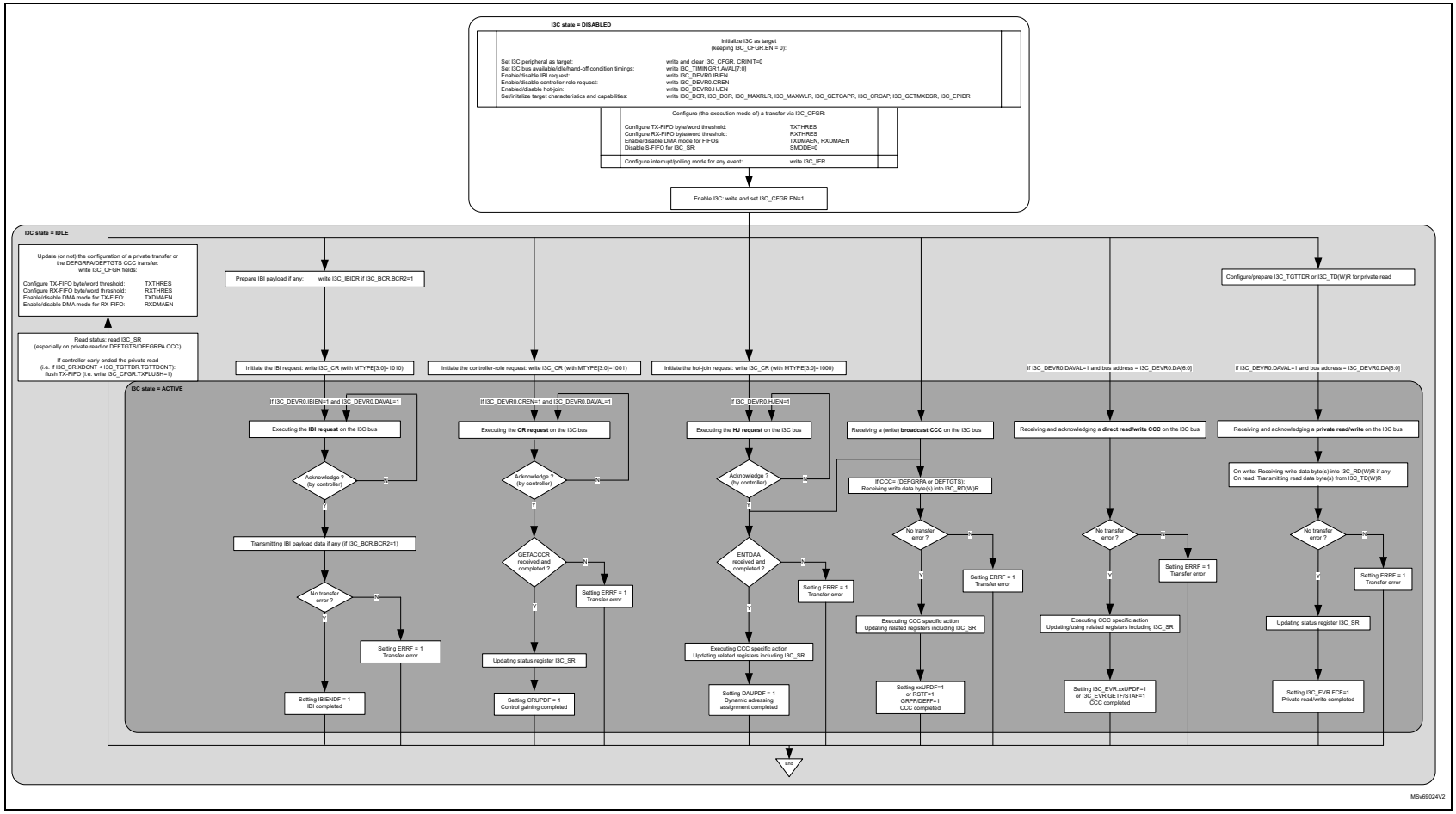
The registers usage versus the I3C peripheral role as controller is summarized in *Section 46.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 483*.

46.7.3 I3C target state and programming sequence

Figure 583 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 583. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message from the controller

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 46.9*:

- *Figure 587: I3C CCC messages, as target*
- *Figure 588: I3C broadcast ENTDAA CCC, as target*
- *Figure 589: I3C broadcast DEFTGTS CCC, as target*
- *Figure 590: I3C broadcast DEFGRPA CCC, as target*
- *Figure 592: I3C private read/write messages, as target*

Figure 583 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 46.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 46.9*:

- *Figure 594: IBI transfer, as controller/target*
- *Figure 595: Hot-join request transfer, as controller/target*
- *Figure 596: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC) from the controller

The registers usage vs. the I3C peripheral role as target is summarized in *Section 46.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 484*.

46.8 I3C registers and programming

46.8.1 I3C register set, as controller/target

Table 482 lists the registers and their usage versus the I3C peripheral role.

Table 482. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_MISR	X	X

Table 482. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVR0	X	X
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

46.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 483* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 483. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 483. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN TRIGHWEN, TRIGPOL and TRIGSEL[3:0] EN(1)	Write: any used field except {CRINIT, HKSDAEN, TRIGHWEN, TRIGPOL and TRIGSEL[3:0]}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) FCFDIS CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 483. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_MISR	X	-	-	Read (controller-role fields): HJMIS CRMIS IBIMIS FCMIS ERRMIS RXTGTENDMIS RXFNEMIS TXFNFMIS SFNEMIS CFNFMIS
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			

Table 483. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

46.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 484* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 484. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 484. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_MISR	X			Read (target-role fields): GRPMIS DEFMIS INTUPDMIS ASUPDMIS RSTMIS MRLUPDMIS MWLUPDMIS DAUPDMIS STAMIS GETMIS WKPMIS CRUPDMIS IBIENDMIS ERRMIS FCMIS RXFNEMIS TXFNFMIS

Table 484. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

46.9 I3C bus transfers and programming

46.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 485*.

Table 485. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 589*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X		
SETAASA	0x29		No defining/sub-command byte	No data byte	X		
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 590*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 485. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF	Update I3C_DEVR0: DA[6:0] (and set DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 46.16.20*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 46.16.19*.

Table 485. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 46.16.29*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 46.16.24*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 46.16.25*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 46.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 46.16.28*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 46.16.26*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 46.16.27*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

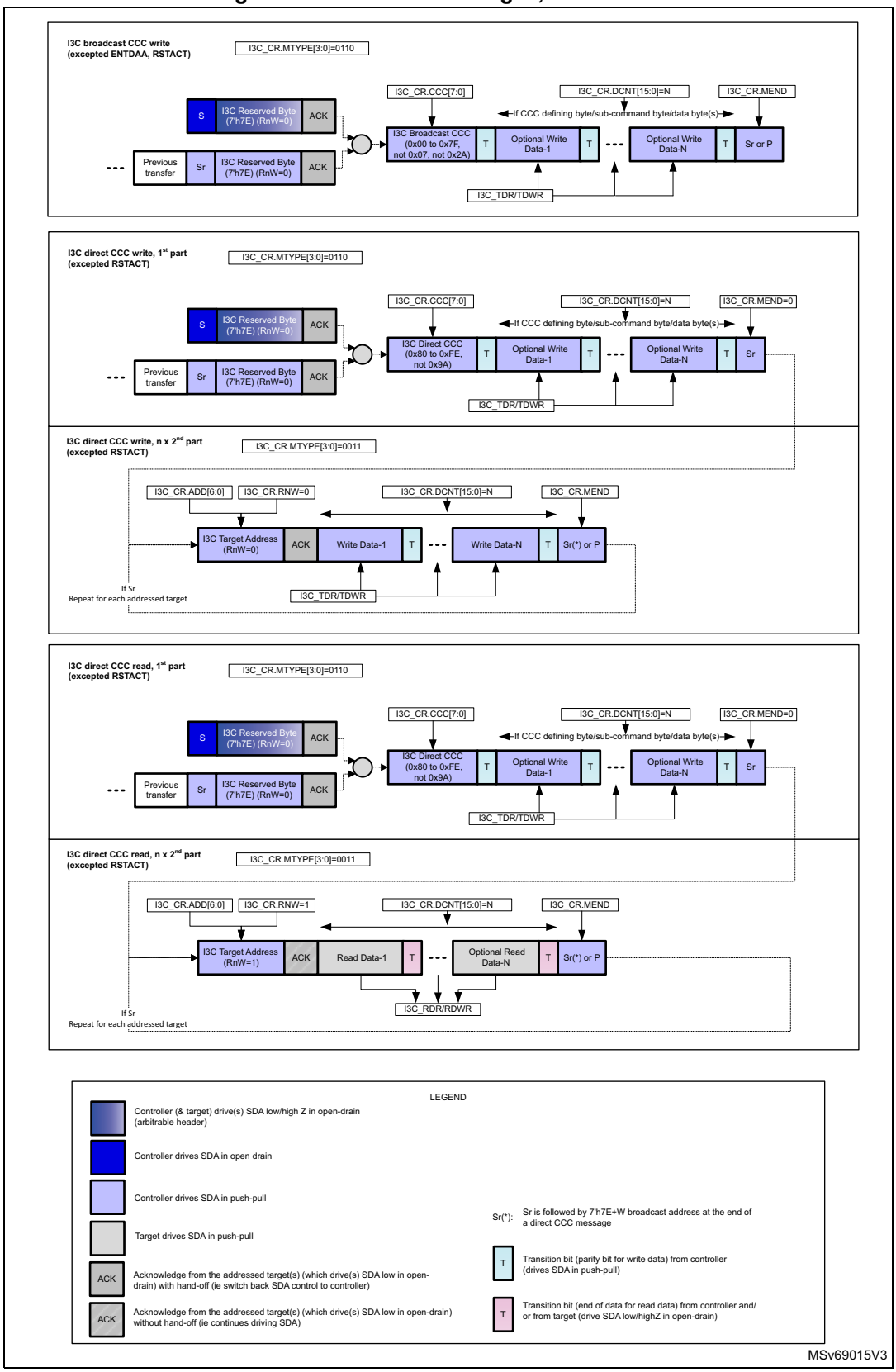
Table 485. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

46.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 584 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

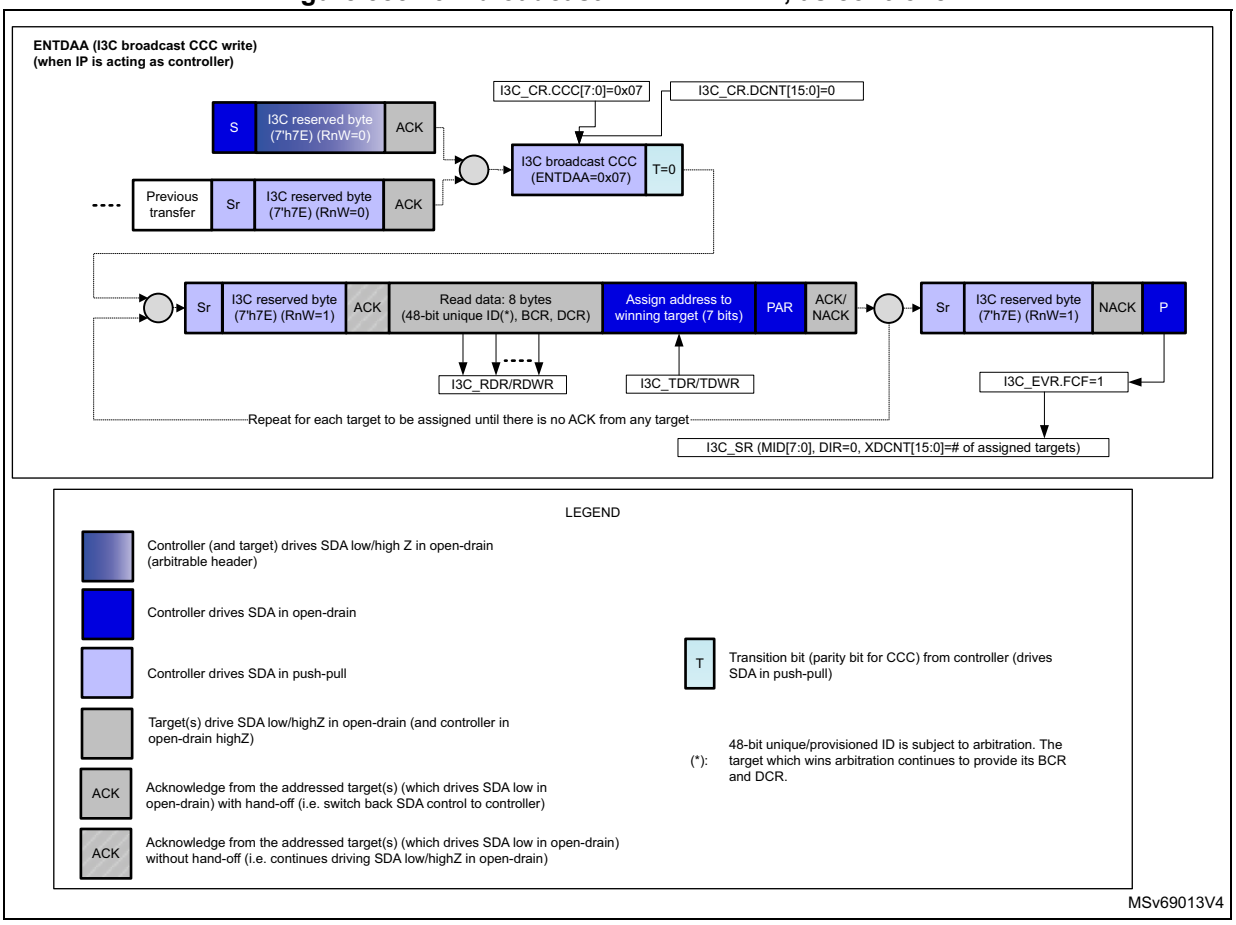
Figure 584. I3C CCC messages, as controller



46.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 585 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

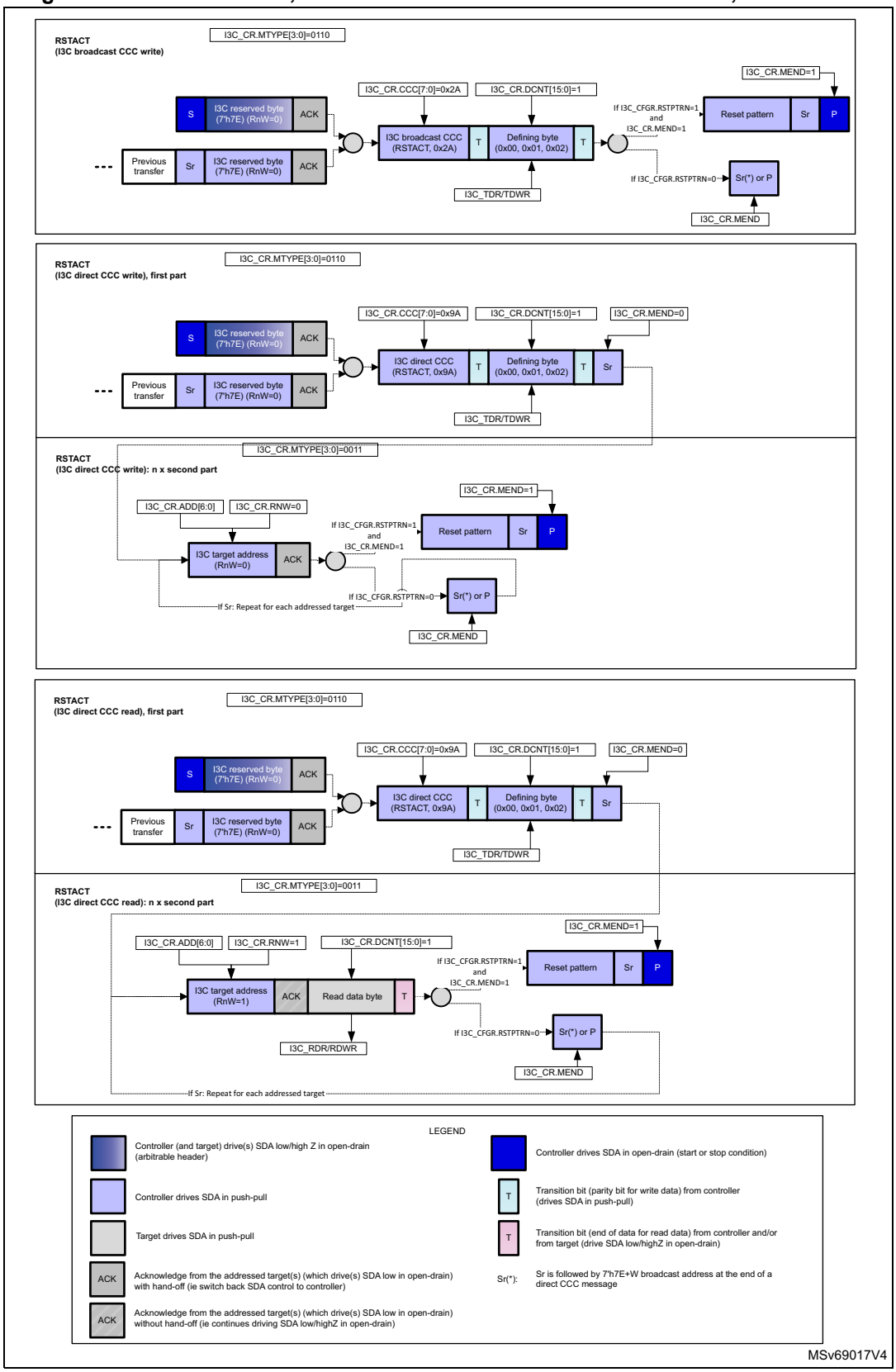
Figure 585. I3C broadcast ENTDAA CCC, as controller



46.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 586 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

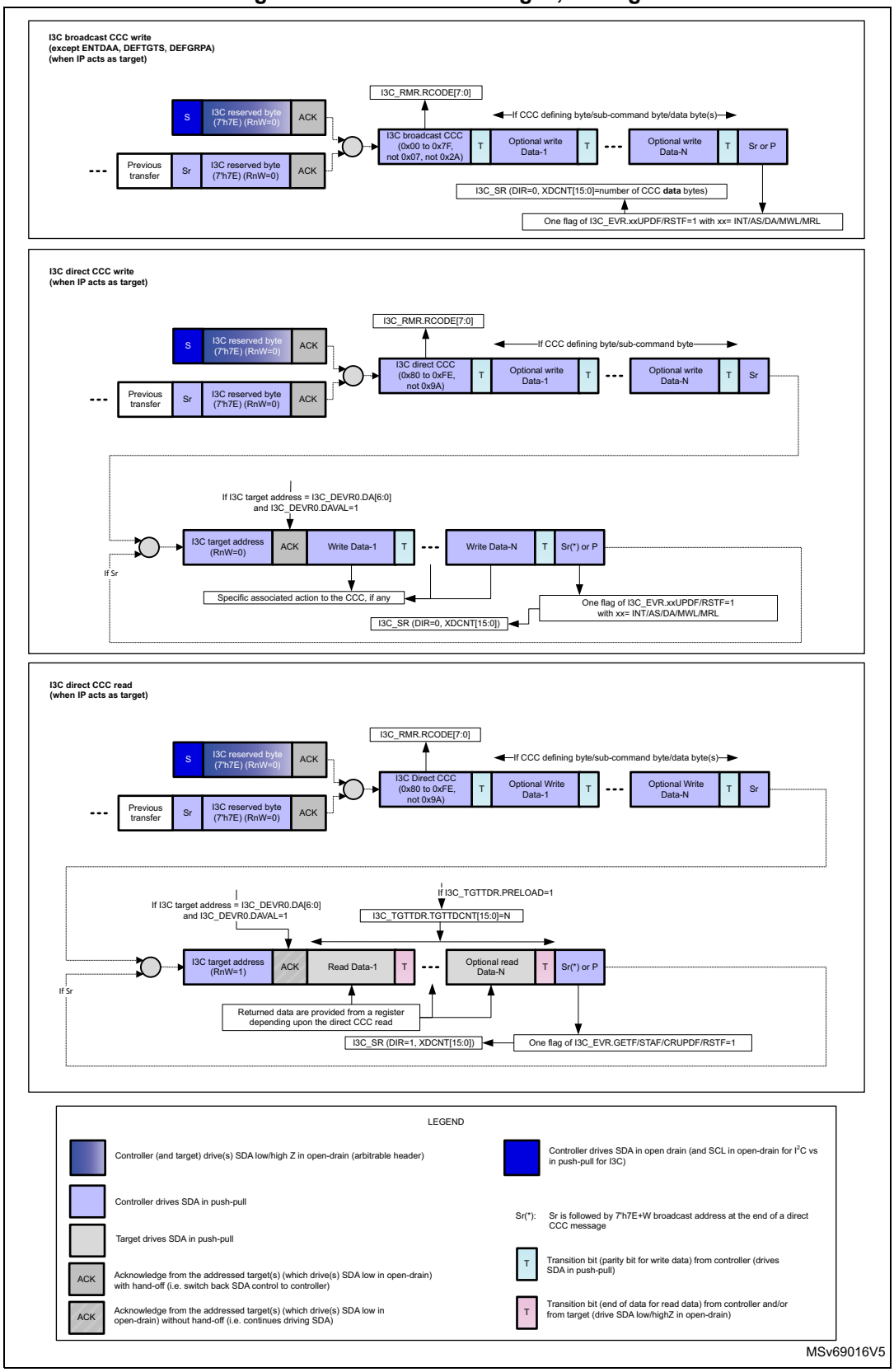
Figure 586. I3C broadcast, direct read and direct write RSTACT CCC, as controller



46.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 587 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

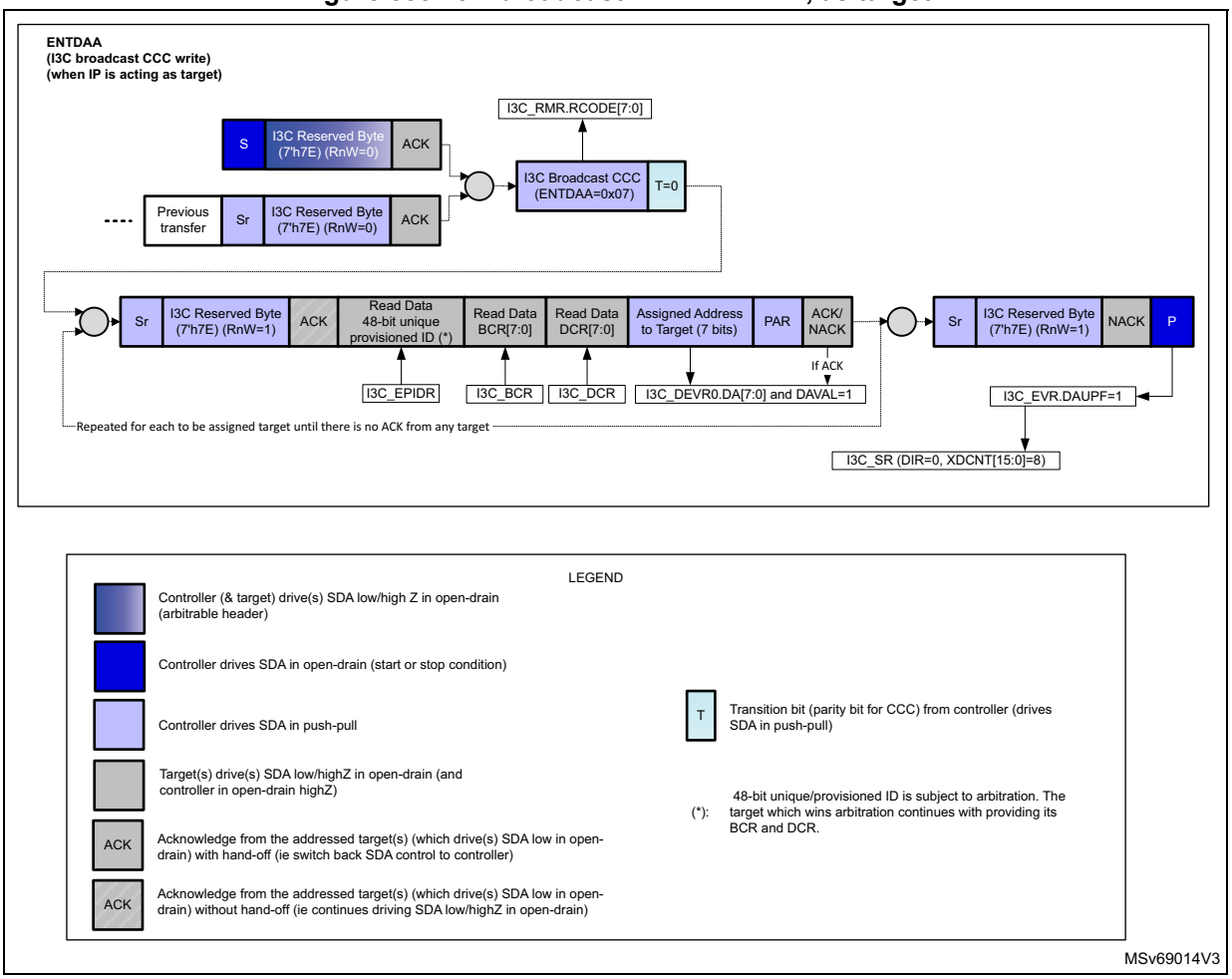
Figure 587. I3C CCC messages, as target



46.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 588 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

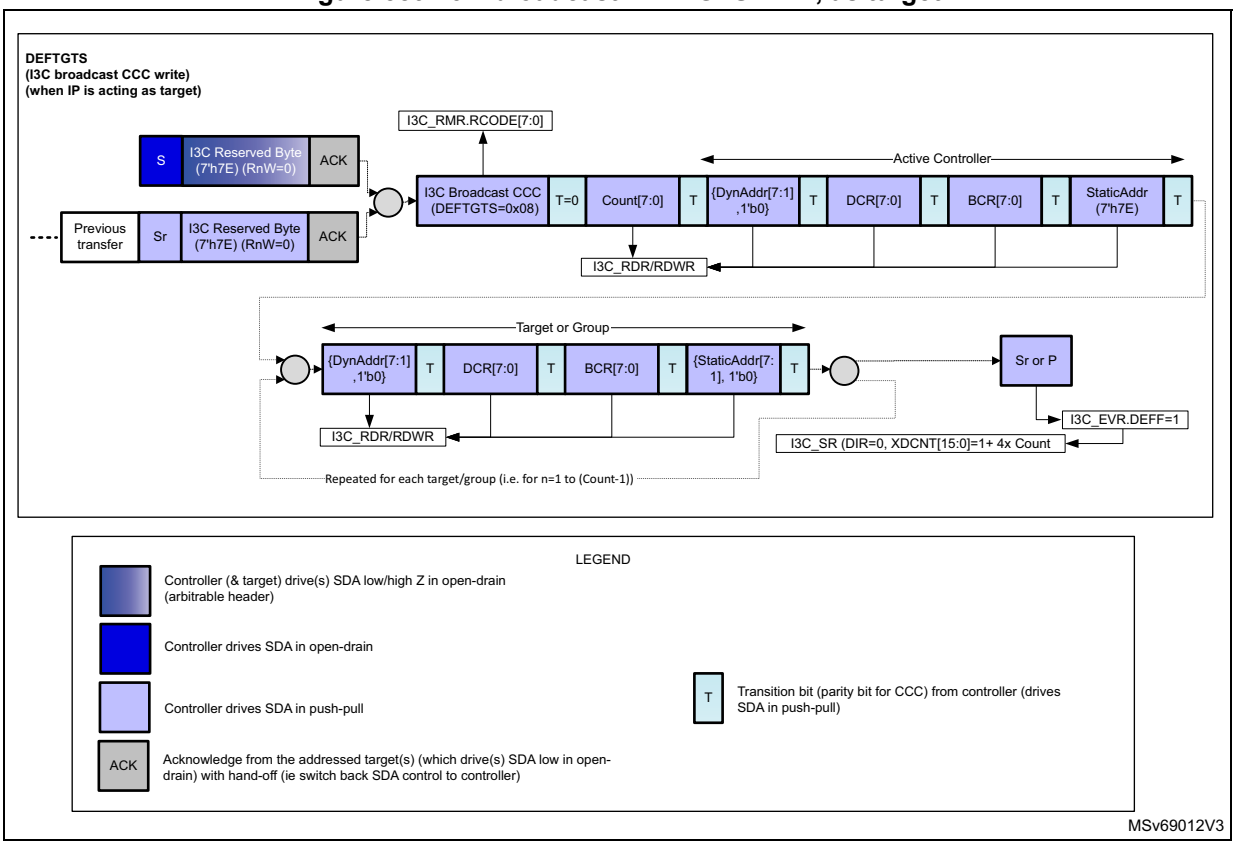
Figure 588. I3C broadcast ENTDAA CCC, as target



46.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 589 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

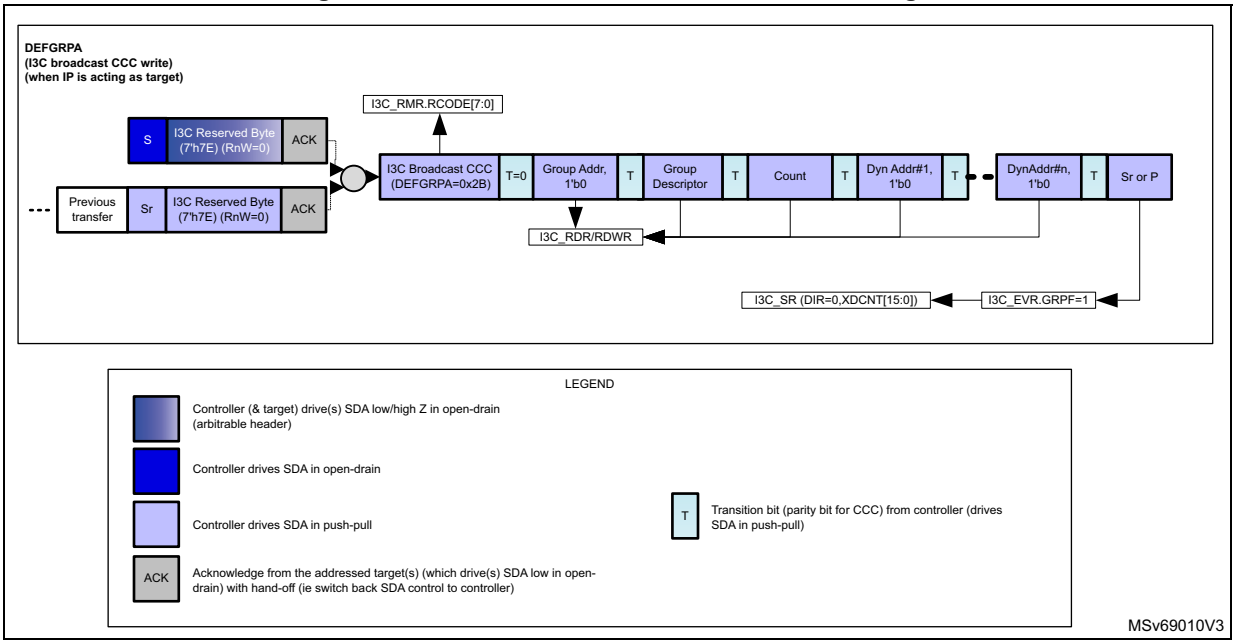
Figure 589. I3C broadcast DEFTGTS CCC, as target



46.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 590 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 590. I3C broadcast DEFGRPA CCC, as target



46.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

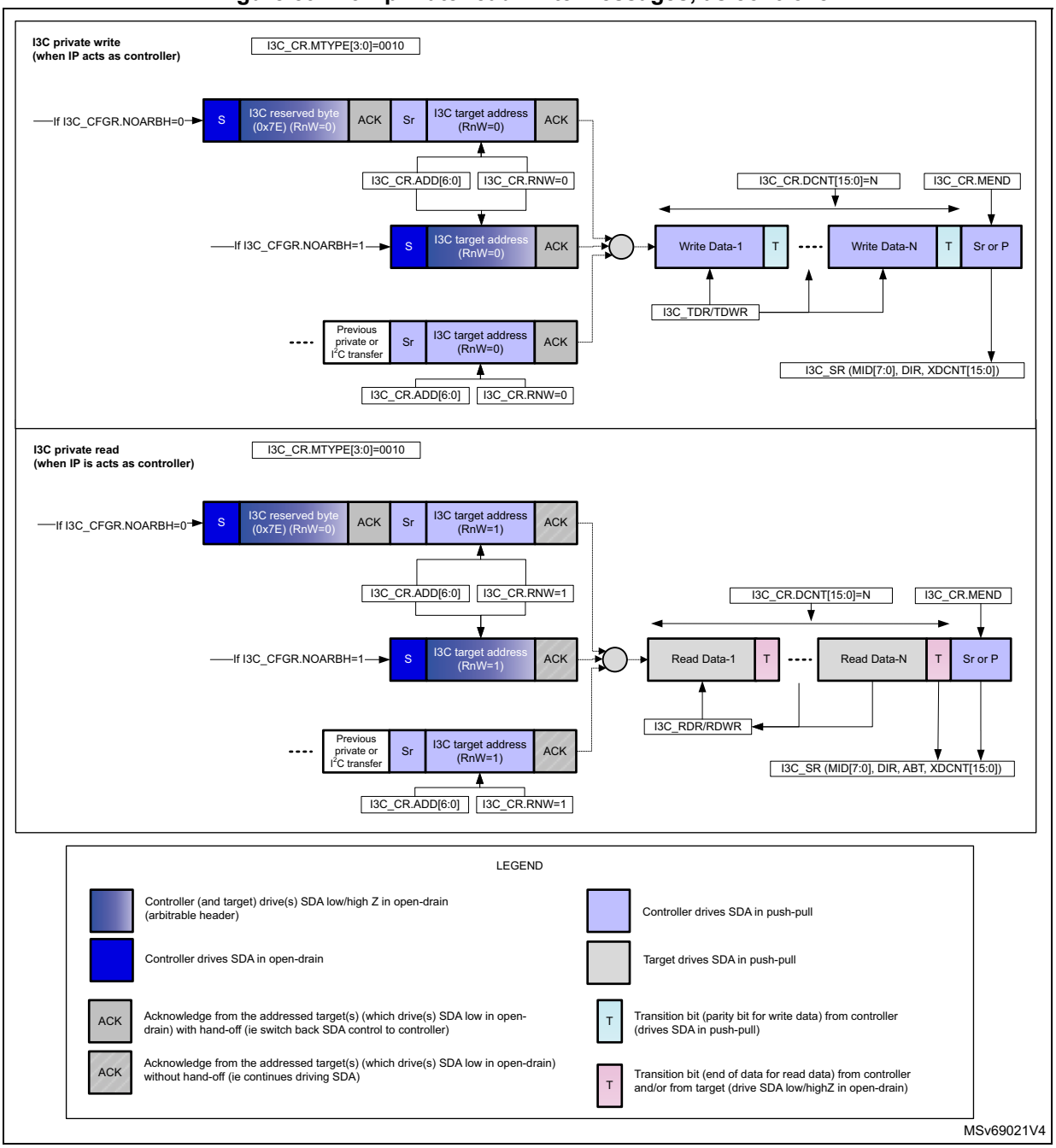
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

46.9.10 I3C private read/write transfer, as controller

Figure 591 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

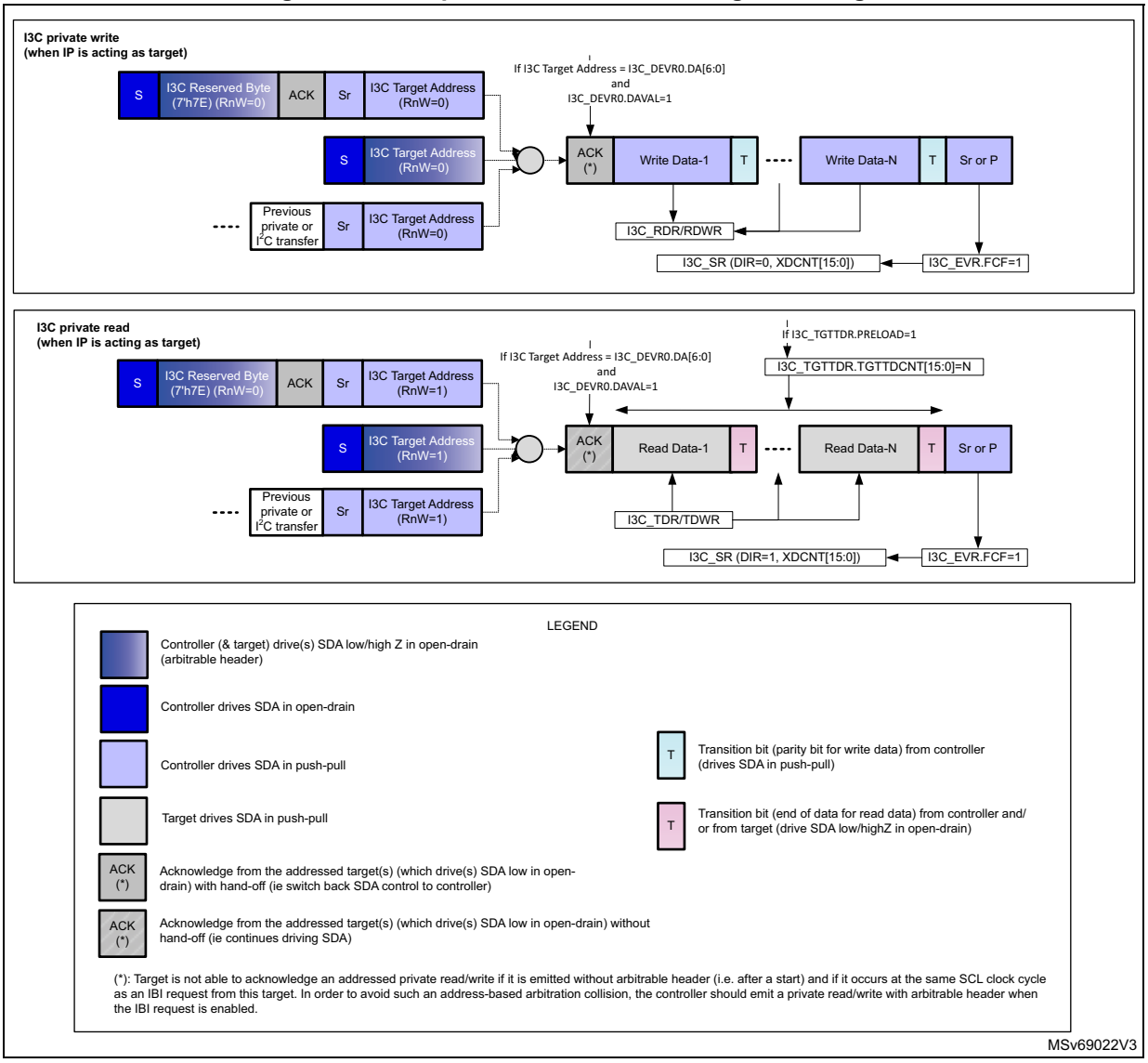
Figure 591. I3C private read/write messages, as controller



46.9.11 I3C private read/write transfer, as target

Figure 592 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

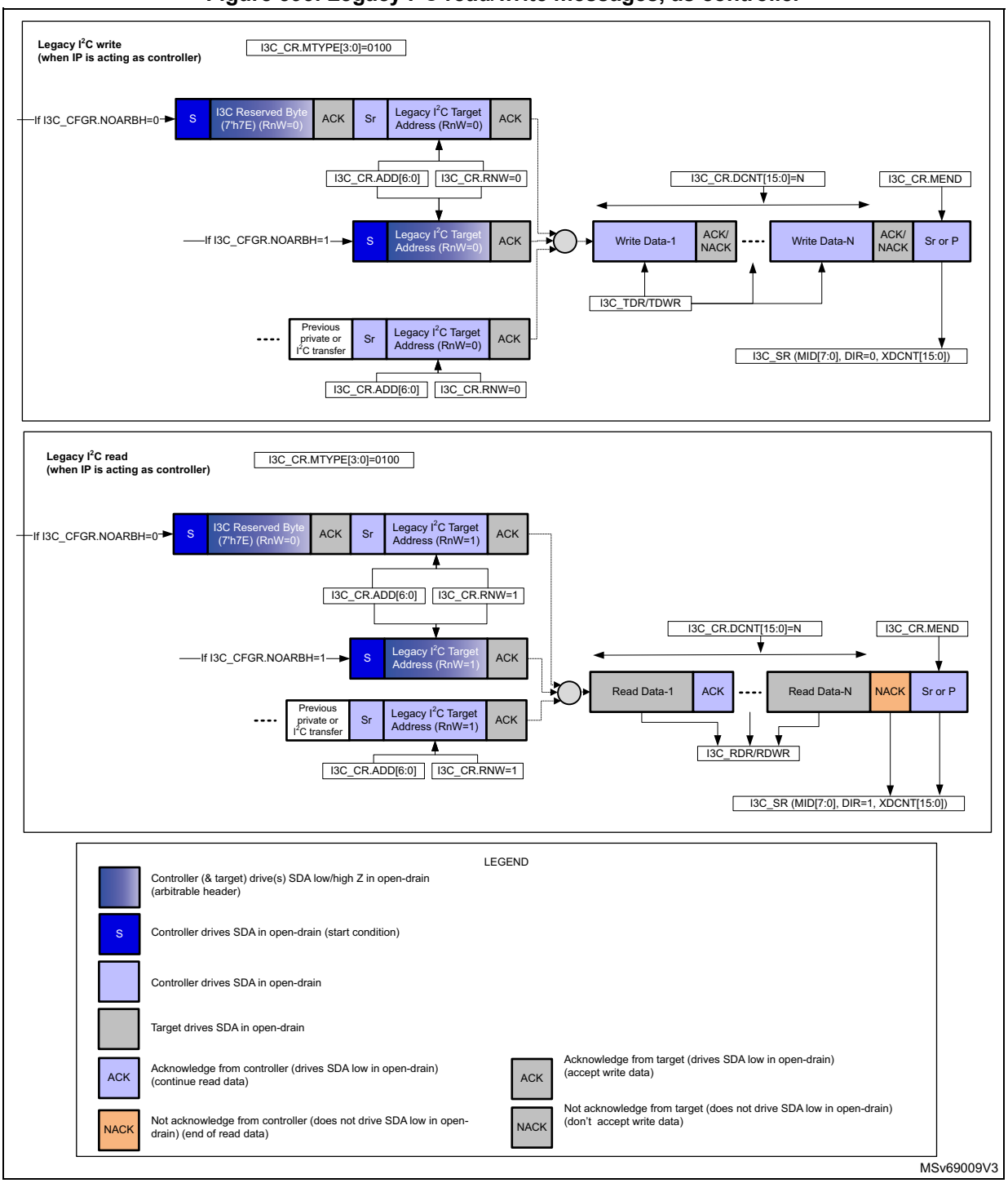
Figure 592. I3C private read/write messages, as target



46.9.12 Legacy I²C read/write transfer, as controller

Figure 593 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

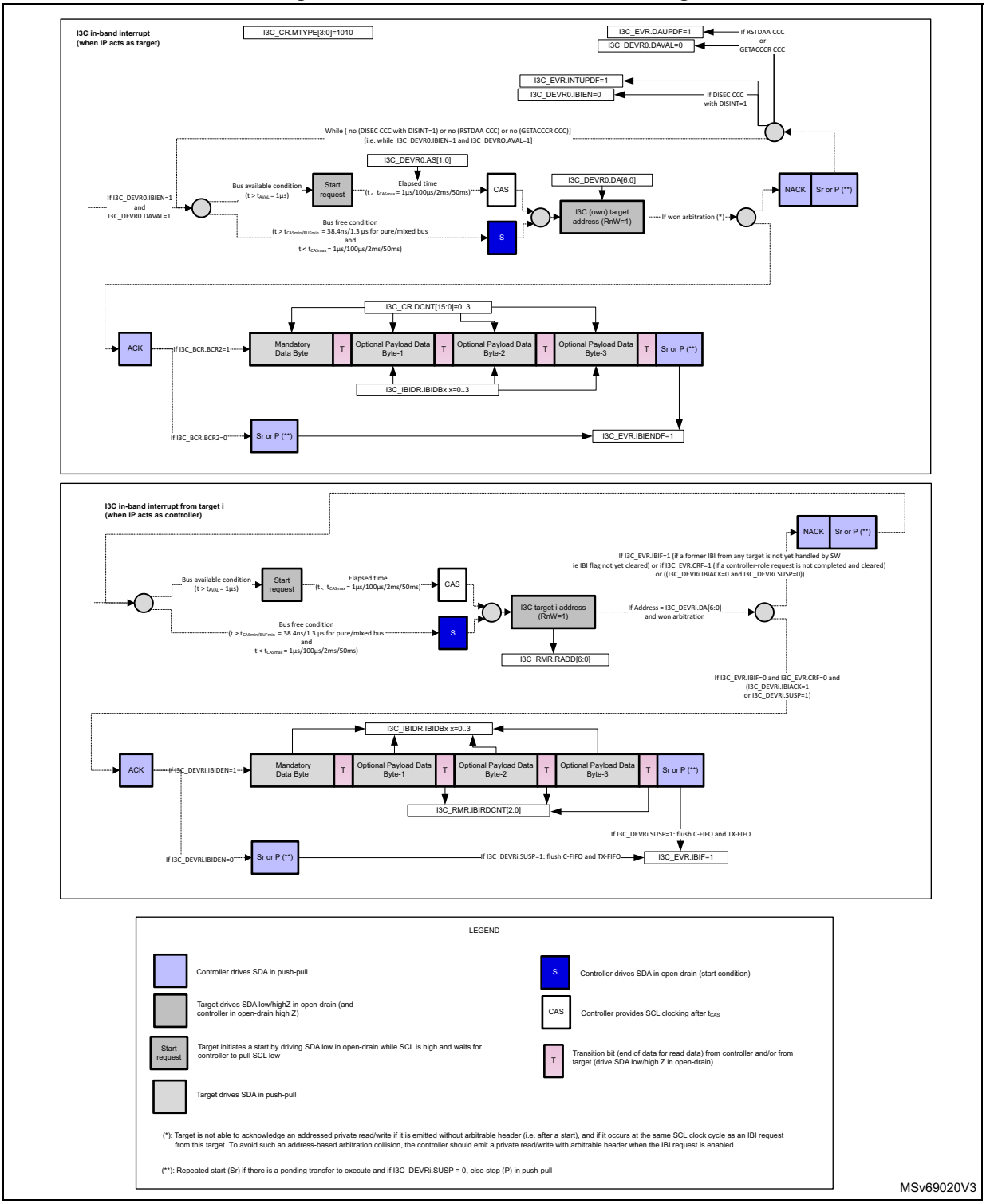
Figure 593. Legacy I²C read/write messages, as controller



46.9.13 I3C IBI transfer, as controller/target

Figure 594 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 594. IBI transfer, as controller/target



When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

Timing control and sensor acquisition timestamping when I3C controller

When the I3C peripheral acts a controller, it can use timing control and asynchronous mode 0 to timestamp the moment at which the target(s) acquired sampled data, and to time-correlate them.

To do this, the software must go through the following sequence:

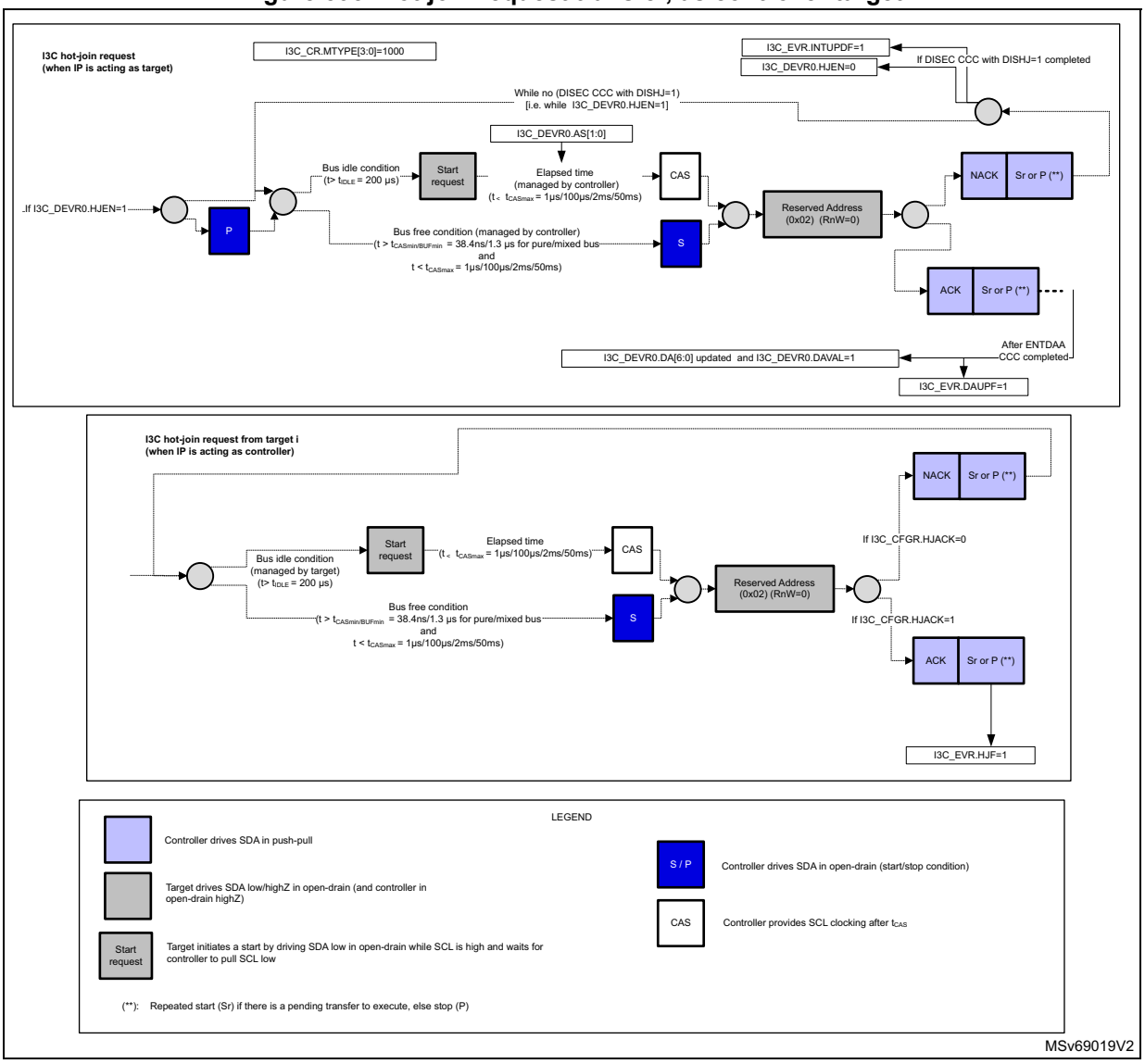
1. Allocate and configure a timer in capture mode, to be connected to the I3C generated event i3c_ibi_ack so that for each IBI acknowledge the counter of the timer can be captured.
2. Program a broadcast/direct SETXTIME CCC with sub-command byte 0xDT to enter asynchronous mode 0 on the target(s) side.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR register), reset and enable the timer to start the counter on the controller side.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF = 1 in the I3C_EVR register), and if the IBI is with timing info (if IBIDB0[7:5] = 100 in the I3C_IBIDR register), read the captured value of the timer, as C_TIM. Given that:
   a) C_C2 = 9 × (SCLL_PP[7:0] + 1 + SCLH_I3C[7:0] + 1) × T_{I3CCLK}
   b) T_C1 (16-bit value) = {IBIDB2, IBIDB1} in the I3C_IBIDR register
   c) T_C2 (8-bit value) = IBID3 in the I3C_IBIDR register

   Compute the timestamp of the sampled data received from the target, as follows (see MIPI spec formula for more details): C_TS = C_TIM - C_C2 x T_C1 / T_C2.
5. Repeat step *4*, for each received IBI.
6. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit asynchronous mode 0 and disable/deallocate the timer resource.

46.9.14 I3C hot-join request transfer, as controller/target

Figure 595 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

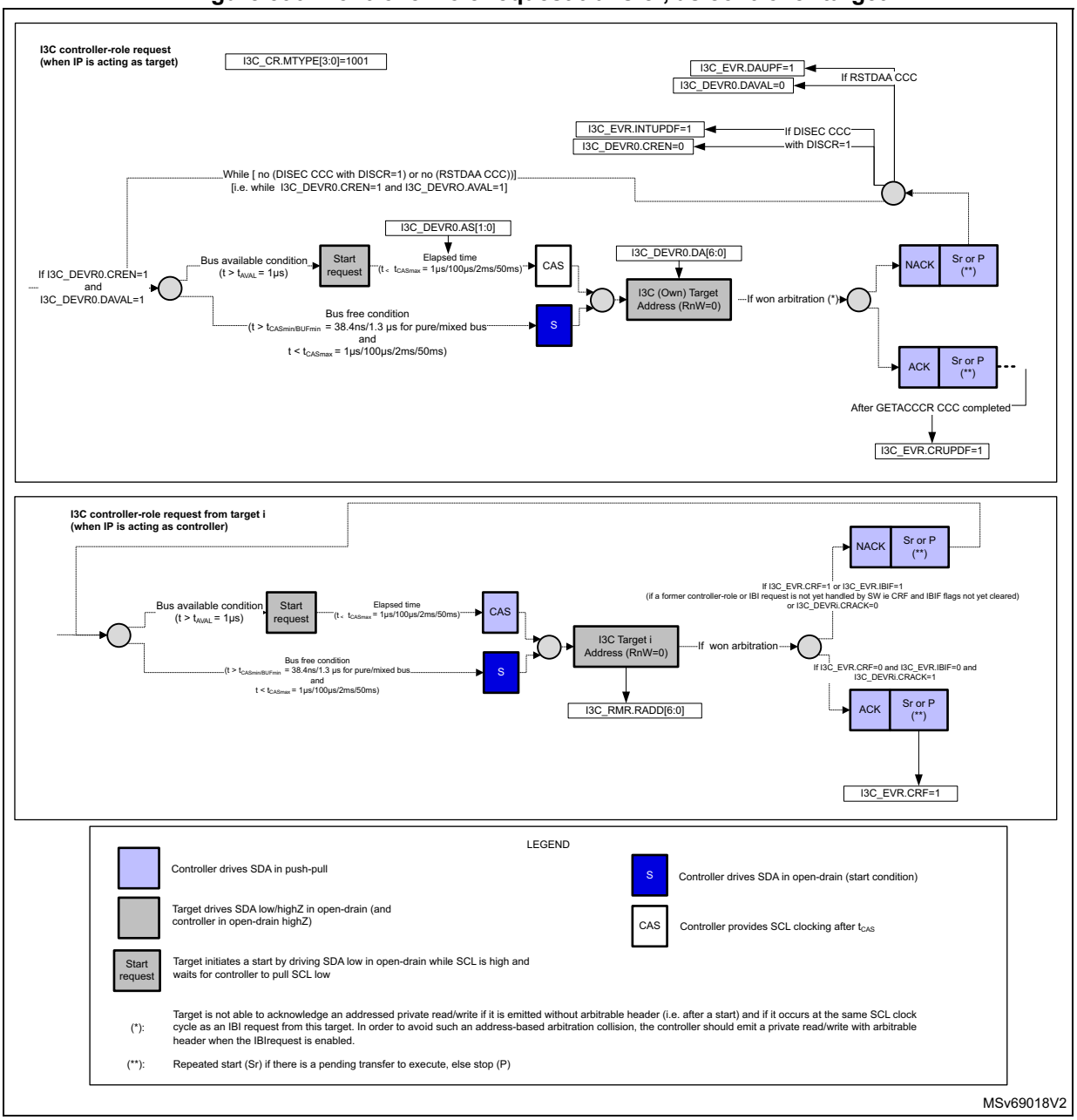
Figure 595. Hot-join request transfer, as controller/target



46.9.15 I3C controller-role request transfer, as controller/target

Figure 596 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 596. Controller-role request transfer, as controller/target



46.10 I3C FIFOs management, as controller

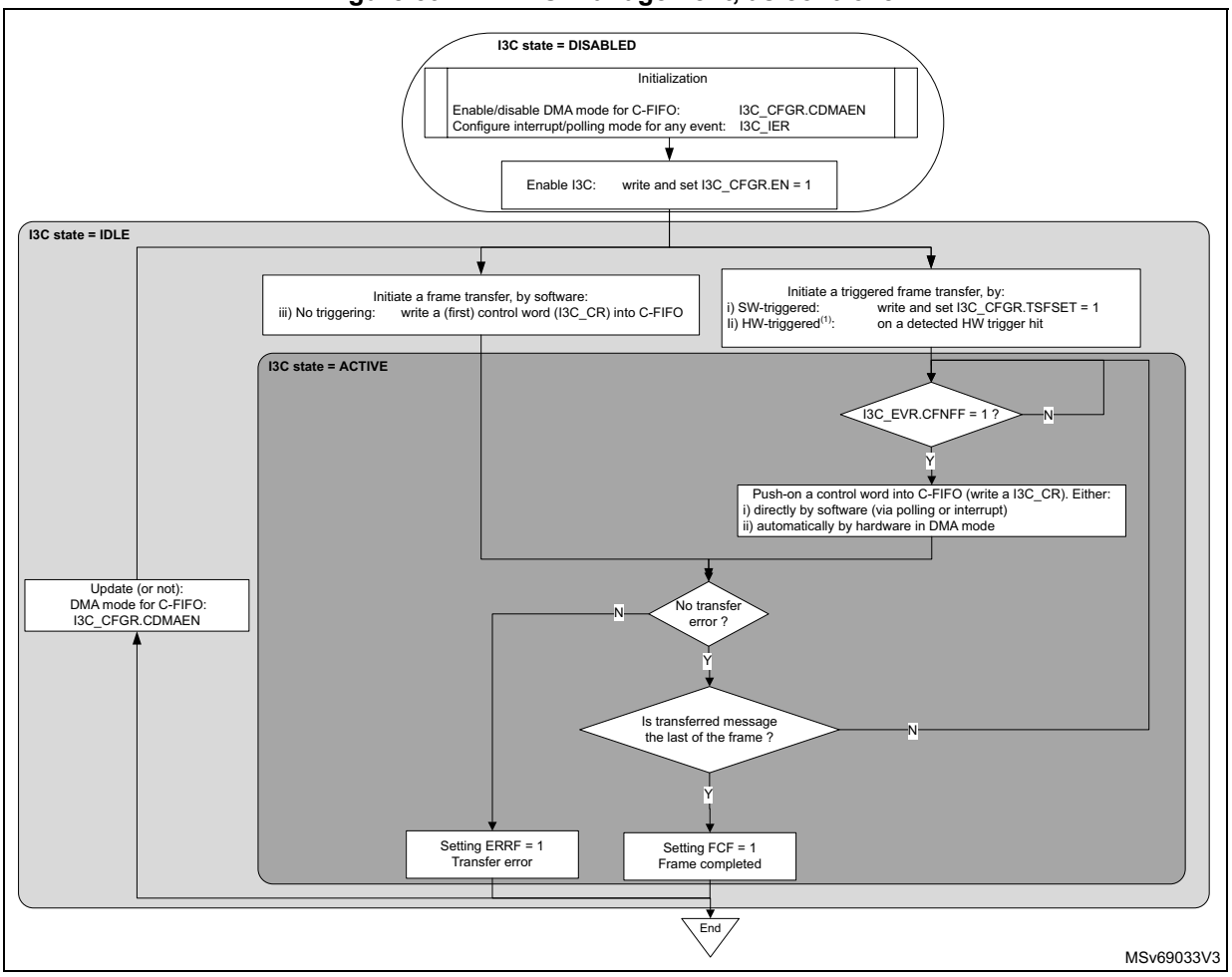
46.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 46.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 584*, *Figure 585*, *Figure 586*)
- direct read/write CCC (*Figure 586*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 591*)
- legacy I2C read/write (*Figure 593*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 597 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 597. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 46.3.5*.

Note: The software can disable the generation of FCF flag by writing 1 into the FCFDIS bit of the I3C_CFGR register. This can be used with hardware triggers and autonomous DMA (if this feature is available, see Section 46.3) to stream a series of frames.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

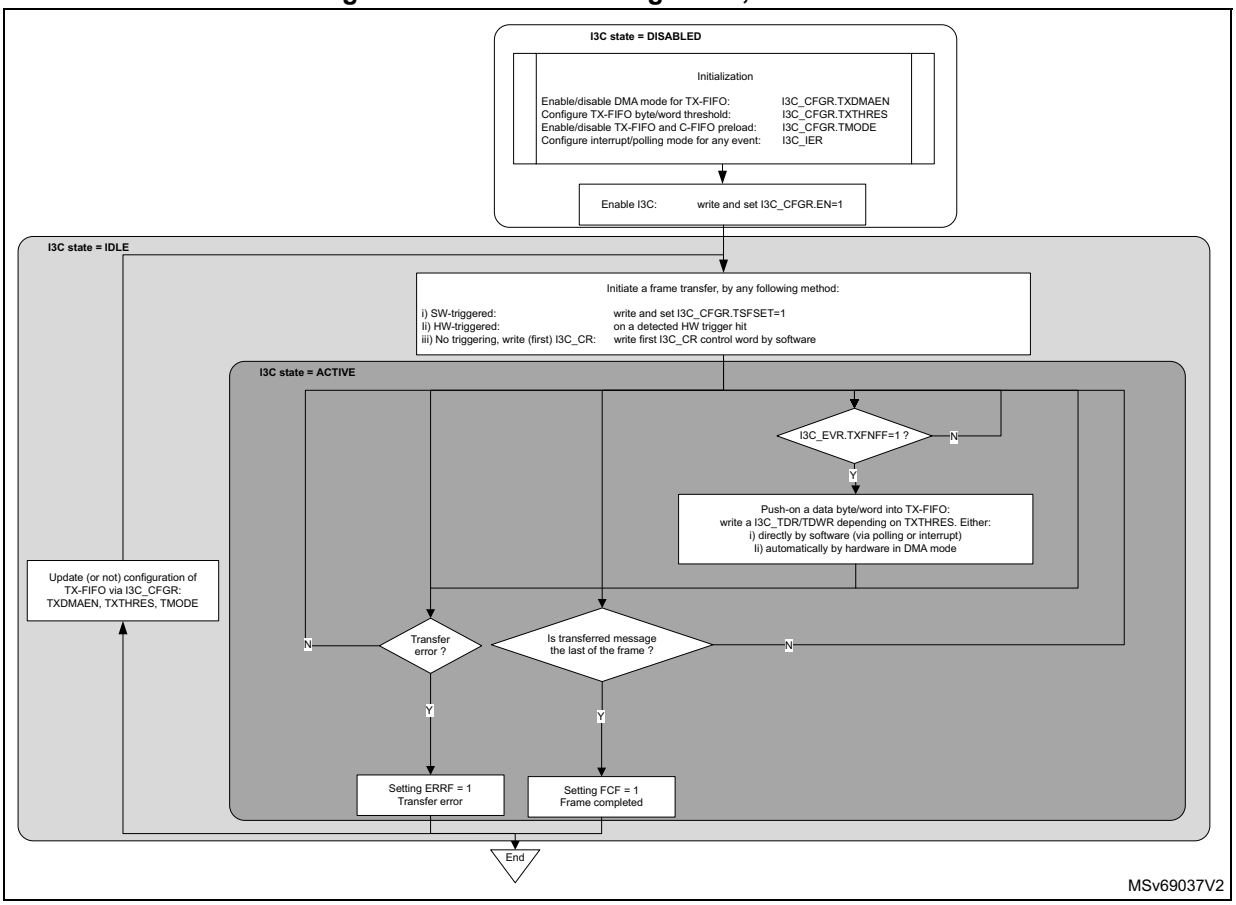
46.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 46.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 584*, *Figure 585*, *Figure 586*)
- private write (*Figure 591*)
- legacy I2C write (*Figure 593*)

Figure 598 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 598. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 476*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

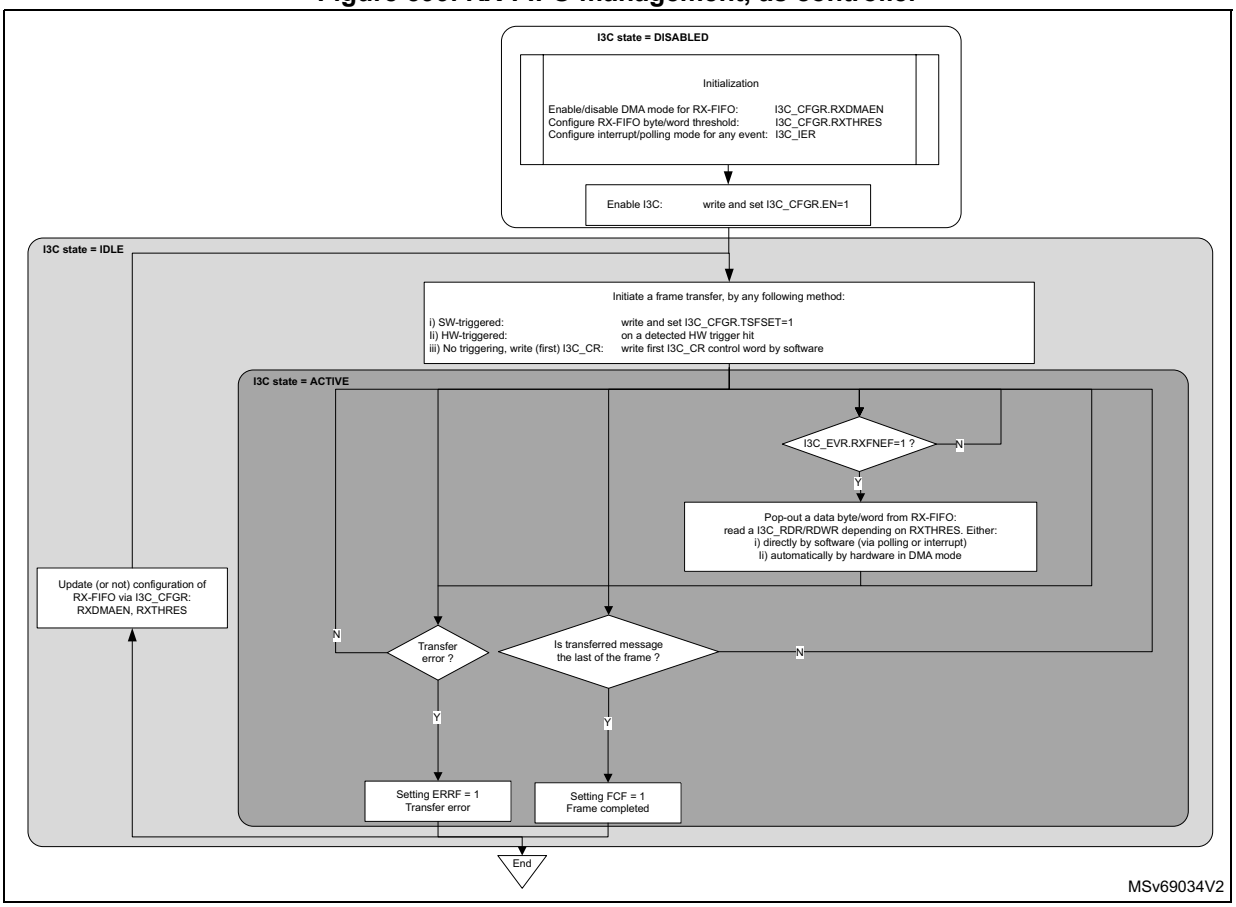
46.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 46.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 585*)
- direct CCC read (*Figure 584*), including direct RSTACT CCC read (*Figure 586*)
- private read (*Figure 591*)
- legacy I2C read (*Figure 593*)

Figure 599 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 599. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 46.10.4*.

46.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

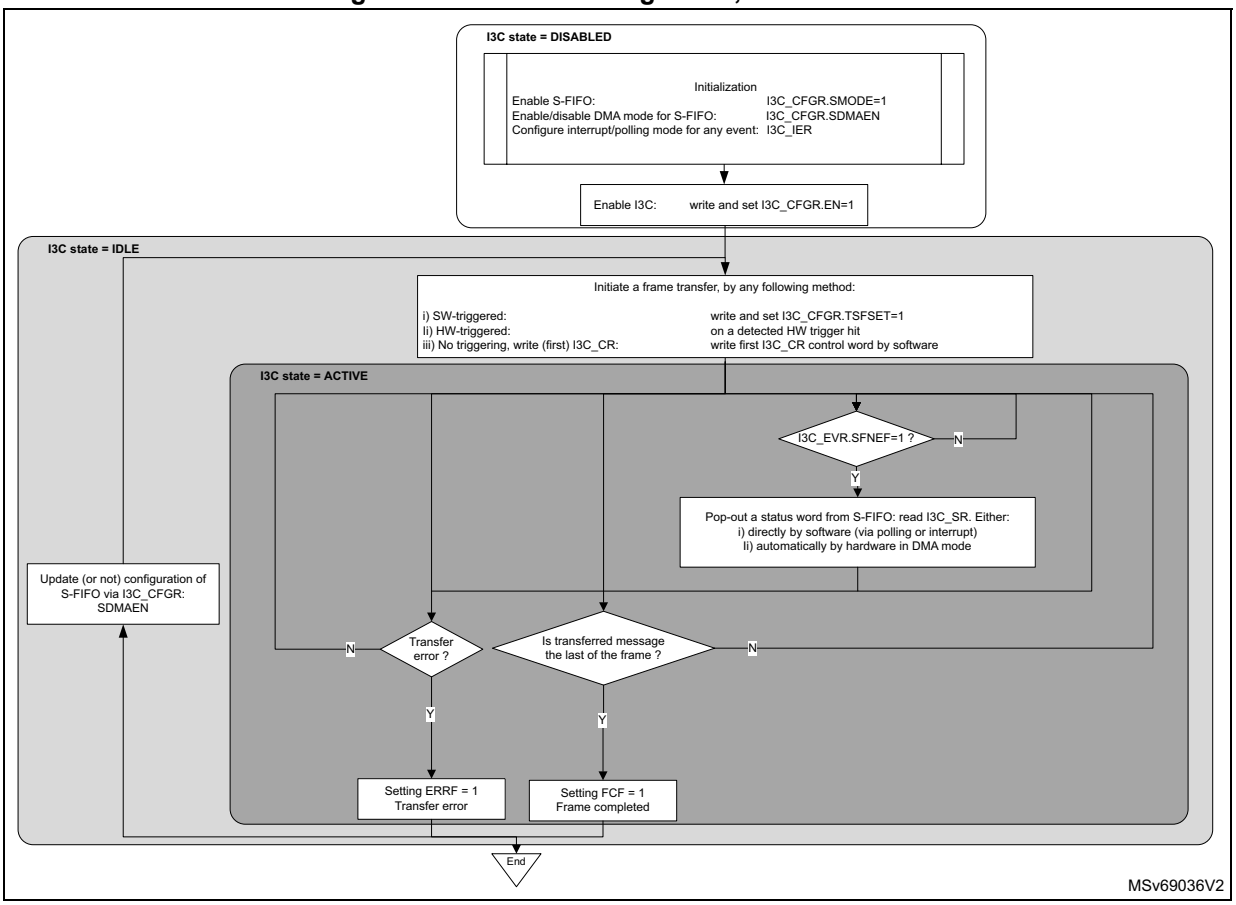
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 46.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 600* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 600. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

46.11 I3C FIFOs management, as target

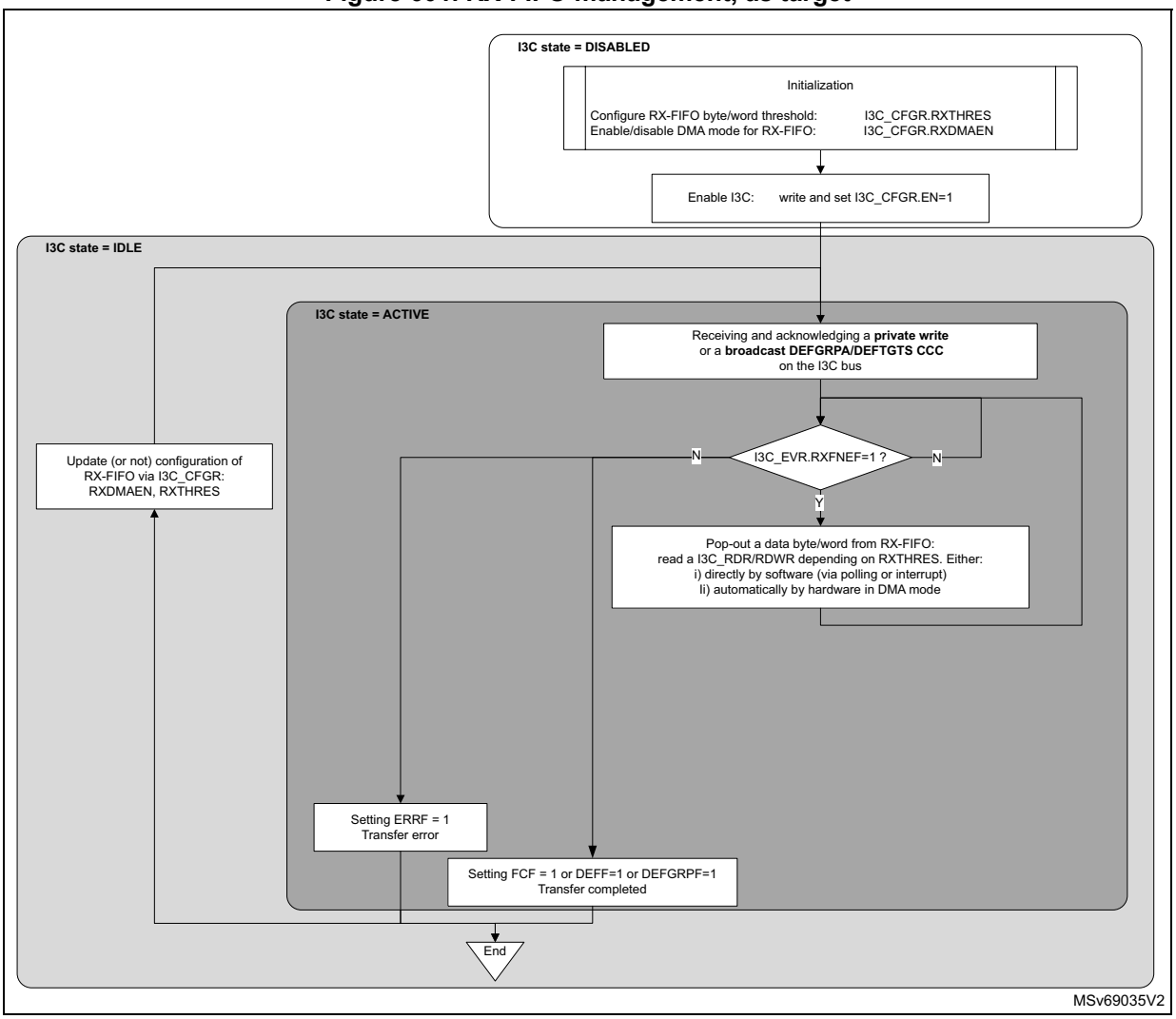
46.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 46.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 589*)
- broadcast DEFGRPA CCC (*Figure 590*)
- private write (*Figure 592*)

Figure 601 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as target.

Figure 601. RX-FIFO management, as target



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

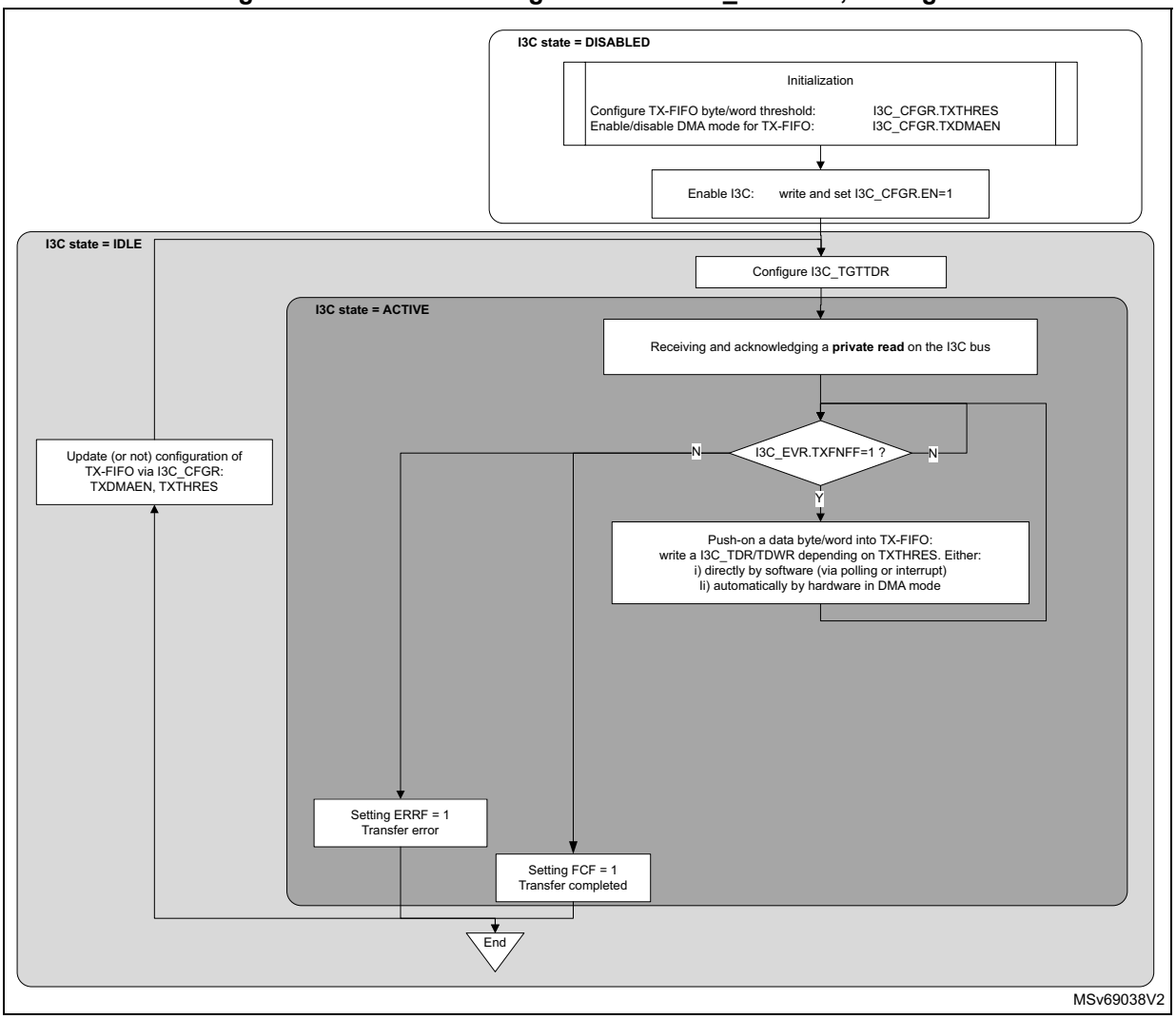
46.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 46.9*, the TX-FIFO is used only during a private read (*Figure 592*).

Figure 602 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 602. TX-FIFO management with I3C_TGTTDR, as target



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus, the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the I3C bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 476).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the I3C bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the I3C bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the I3C bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another private read, depending upon the user application

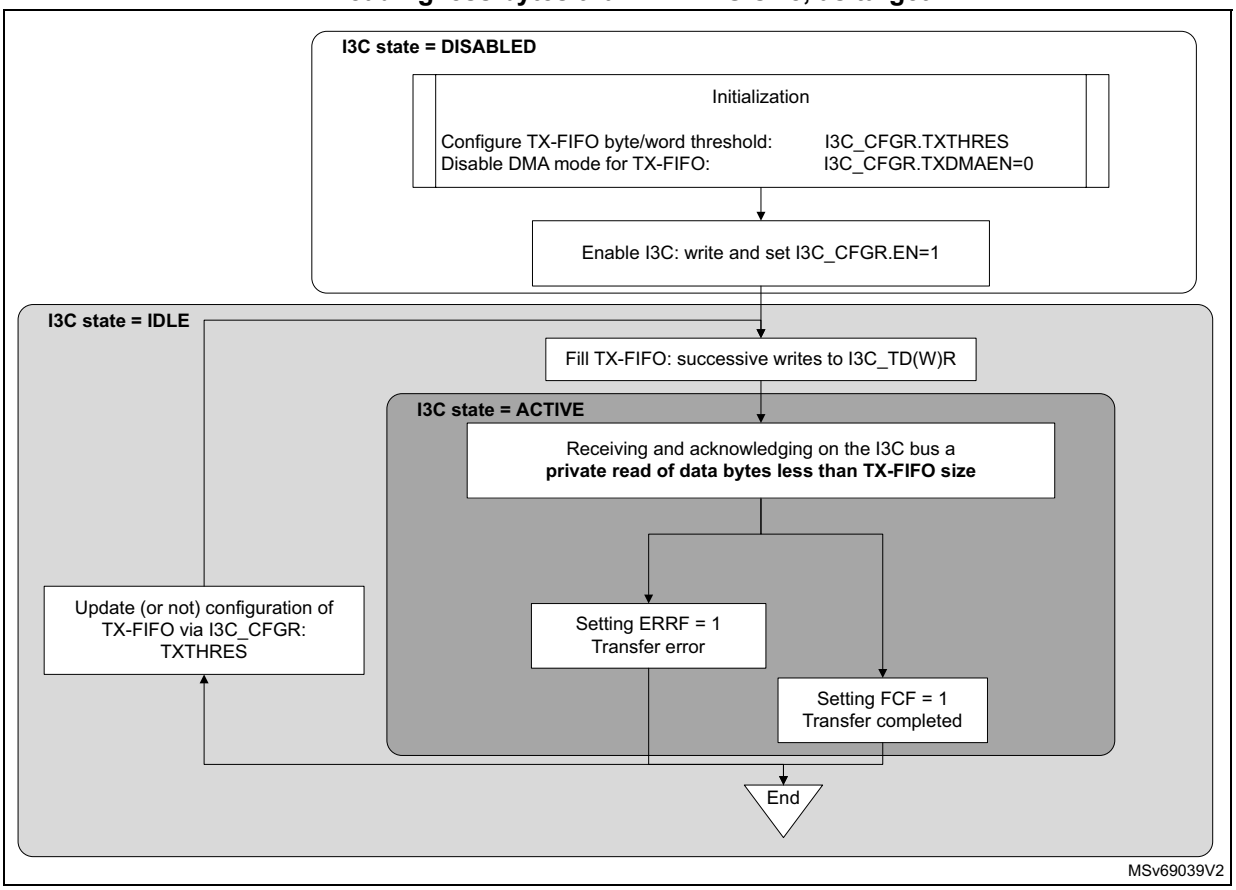
If TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 603*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the I3C bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 603. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



46.12 I3C error management

46.12.1 Controller error management

Table 486 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 486. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 486. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 486. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

46.12.2 Target error management

Table 487 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 487. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 487. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error	SDA is detected as driven at unexpected value on an SDR data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

46.13 I3C wake-up from low-power mode(s)

46.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to *Section 10: Reset and clock control (RCC)*.

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x=1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
   - On a detected hardware trigger to initiate an I3C transfer on the I3C bus.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 594*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 595*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 596*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.
7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high)
   - As target, as initiated by an external I3C device, whatever controller or target.
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a I3C bus start is missed and waits for the kernel clock to be provided. It can be noted that an I3C controller message intended to be addressed to it may have been missed (and NAcked). Or may have been missed any I3C CCC or private message from the controller to another addressed target or an IBI/CR/HJF start request from another target.
   - On a detected hardware trigger to initiate an I3C IBI/CR/HJ transfer on the I3C bus
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that an I3C bus transaction was missed, but not whether the target was addressed or not. The

software should use a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

46.13.2 52.13.1 I3C wake-up from Standby

As target: wake-up on a reset pattern

46.14 I3C in low-power modes

Table 488. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode. I3C transfers can occur and can be assisted with an autonomous DMA for data transfers to/from memory, provided that the autonomous DMA can operate in Stop mode. Refer to *Section 46.13* for more details.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode. The I3C reset pattern detection module in the V_{DD} domain can wake up the device from Standby, if enabled via the PWR module.

1. Refer to *Table 474* and *Table 475* to know if Stop mode is supported, and which one.

46.15 I3C interrupts

Table 489. I3C interrupt requests

Interrupt acronym		Interrupt event	Used as		Interrupt enable: field in I3C_IER	Event flag: field in I3C_EVR	Masked interrupt status: field in I3C_MISR	Event clear method: write 1 to field in I3C_CEVR
			Controller	Target				
I3C	EVT	A control word is requested	X	-	CFNFIE	CFNFF	CFNFMIS	-
		A status word is available	X	-	SFNEIE	SFNEF	SFNEMIS	-
		Data to be transmitted are requested	X	X	TXFNFIE	TXFNFF	TXFNFMIS	-
		Received data are available	X	X	RXFNEIE	RXFNEF	RXFNEMIS	-
		Controller: frame transfer completed Target: private transfer completed	X	X	FCIE	FCF	FCMIS	CFCF
		A private read transfer prematurely ended by the target (and SMODE = 0 in the I3C_CFGR register)	X	-	RXTGTENDIE	RXTGTENDF	RXTGTENDMIS	CRXTGTENDF
		An IBI request is received	X	-	IBIIE	IBIF	IBIMIS	CIBIF
		A controller-role request is received	X	-	CRIE	CRF	CRMIS	CCRF
		A hot-join request is received	X	-	HJIE	HJF	HJMIS	CHJF
		IBI request is completed	-	X	IBIENDIE	IBIENDF	IBIENDMIS	CIBIENDF
		I3C bus start is missed	-	X	WKPIE	WKPF	WKPMIS	CWKPF
		Direct GETACCR CCC is received	-	X	CRUPDIE	CRUPDF	CRUPDMIS	CCRUPDF
		Direct GETSTATUS CCC is received	-	X	STAIE	STAF	STAMIS	CSTAF
		Any direct GETxxx CCC (except GETSTATUS) is received	-	X	GETIE	GETF	GETMIS	CGETF
		Dynamic address is updated (broadcast ENTDAA or RSTDAA, or direct SETNEWDA is received)	-	X	DAUPDIE	DAUPDF	DAUPDMIS	CDAUPDF
		Direct SETMWL CCC is received	-	X	MWLUPDIE	MWLUPDF	MWLUPDMIS	CMWLUPDF
		Direct SETMRL CCC is received	-	X	MRLUPDIE	MRLUPDF	MRLUPDMIS	CMRLUPDF
		A reset pattern is detected	-	X	RSTIE	RSTF	RSTMIS	CRSTF
		Bus activity state is updated (direct/broadcast ENTASx CCC is received)	-	X	ASUPDIE	ASUPDF	ASUPDMIS	CASUPDF
		Broadcast/direct ENEC/DISEC CCC is received	-	X	INTUPDIE	INTUPDF	INTUPDMIS	CINTUPDF
		Broadcast DEFTGTS CCC received	-	X	DEFIE	DEFF	DEFMIS	CDEFF
		Broadcast DEFGRPA CCC received	-	X	GRPIE	GRPF	GRPMIS	CGRPF
	ERR	An error occurred	X	X	ERRIE	ERRF	ERRMIS	CERRF

46.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

46.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 46.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 486*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 486* and MIPI specification about escalation handling.

If the fields of the I3C_CFGR register are programmed with EXITPTRN= 0 and RSTPTRN=1, a target reset pattern is sent on the I3C bus after the header message, whatever the header is ACK-ed or NACK-ed. This can be used to generate and broadcast a target reset pattern, especially in the case of an I3C bus topology with a single I3C target in deep sleep. Alternatively, and more generally, a target reset pattern can be encapsulated within a RSTACT CCC command (refer to *Figure 586*).

0010: **private message** (refer to *Figure 591*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 584*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 593*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 595*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 596*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 594*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.

0: write message

1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)

When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.

When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).

Linear encoding up to 64 Kbytes - 1

0x0000: no data to transfer

0x0001: 1 byte

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

46.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 485*, *Figure 584*, *Figure 585*, *Figure 586*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 585*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 585).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 586).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

46.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET, TRIGHWEN,TRIGPOL, TRIGSEL[3:0]), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH, FCFDIS)

The configuration fields CRINIT, HKSDAEN, TRIGHWEN, TRIGPOL, and TRIGSEL[3:0] can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	TRIG HWEN	TRIG POL	TRIGSEL[3:0]				FCFDIS	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w	rw	rw	rw	rw	rw	rw	rw		w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bit 29 **TRIGHWEN**: Hardware trigger enable (when IP acts as controller)

When I3C acts as controller, this bit defines the trigger mode: hardware-triggered or software-triggered.

This bit can be modified only when the IP is disabled (EN = 0 in the I3C_CFGR register).

0: hardware trigger mode is disabled

1: hardware trigger mode is enabled

Note: In hardware trigger mode, the selected trigger is not monitored meanwhile there is an on-going frame transfer. When there is no on-going frame transfer, the selected hardware trigger is monitored and on a trigger hit, the transfer is initiated by asserting CFNFF.

Bit 28 **TRIGPOL**: Hardware trigger signal polarity (when IP acts as controller)

This bit defines the polarity of the selected trigger signal as defined by TRIGSEL[3:0], for when IP acts as I3C controller and hardware trigger mode is enabled (TRIGHWEN = 1).

This bit can be modified only when the IP is disabled (EN = 0 in the I3C_CFGR register).

0: rising edge

1: falling edge

Bits 27:24 **TRIGSEL[3:0]**: Hardware trigger signal selection (when IP acts as controller)

This bitfield selects the trigger signal as per *Table 477* for when IP acts as I3C controller and hardware trigger mode is enabled (TRIGHWEN = 1).

This bitfield can be modified only when the IP is disabled (EN = 0 in the I3C_CFGR register).

Bit 23 **FCFDIS**: FCF generation disable (when I3C acts as controller)

When the IP acts as controller, the generation of the FCF flag in the register I3C_EVR can be disabled. The software can avoid clearing this flag between two generated I3C frames (by writing 1 into CFCF in the I3C_CEVR register), especially in hardware-triggered autonomous mode.

0: enable the generation of the flag FCF in the I3C_EVR register

1: disable the generation of the flag FCF in the I3C_EVR register

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame

- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.

- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel). In any case, when the IP acts as controller, a frame transfer can be hardware- or software-triggered, depending upon the TRIGHWEN field.

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 46.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.

- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).

- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Additionally, if RSTPTRN = 1 and EXITPTRN = 0, a reset pattern is sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register), whatever the message header is ACK-ed or NACK-ed.

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).

- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.

- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN, TRIGHWEN, TRIGSEL[7:0] and TRIGPOL for the I3C_CFGR)

46.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR

register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

46.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

46.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

46.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a

data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

46.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

46.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE LOAD
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

46.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

46.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

46.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.
								r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

46.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

46.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

46.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

46.16.16 I3C masked interrupt status register (I3C_MISR)

Address offset: 0x05C

Reset value: 0x0000 0000

This is a read register, containing the masked interrupt status for each individual event flag bit. Each bit is the logical AND of the corresponding interrupt enable bit of the I3C_IER register and the event flag bit of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPMIS**: DEFGRPA CCC masked interrupt status (when the I3C acts as target)

Bit 30 **DEFMIS**: DEFTGTS CCC masked interrupt status (when the I3C acts as target)

Bit 29 **INTUPDMIS**: ENEC/DISEC CCC masked interrupt status (when the I3C acts as target)

Bit 28 **ASUPDMIS**: ENTASx CCC masked interrupt status (when the I3C acts as target)

Bit 27 **RSTMIS**: reset pattern masked interrupt status (when the I3C acts as target)

Bit 26 **MRLUPDMIS**: SETMRL CCC masked interrupt status (when the I3C acts as target)

Bit 25 **MWLUPDMIS**: SETMWL CCC masked interrupt status (when the I3C acts as target)

Bit 24 **DAUPDMIS**: ENTDAA/RSTDAA/SETNEWDA CCC masked interrupt status (when the I3C acts as target)

Bit 23 **STAMIS**: format 1 GETSTATUS CCC masked interrupt status (when the I3C acts as target)

Bit 22 **GETMIS**: GETxxx CCC masked interrupt status, except GETSTATUS of format 1 (when the I3C acts as target)

Bit 21 **WKPMIS**: Wake-up/missed start masked interrupt status (when the I3C acts as target)

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJMIS**: Hot-join masked interrupt status (when the I3C acts as controller)

Bit 18 **CRUPDMIS**: Controller-role update masked interrupt status (when the I3C acts as target)

Bit 17 **CRMIS**: Controller-role request masked interrupt status (when the I3C acts as controller)

Bit 16 **IBIENDMIS**: IBI end masked interrupt status (when the I3C acts as target)

Bit 15 **IBIMIS**: IBI request masked interrupt status (when the I3C acts as controller)

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRMIS**: error masked interrupt status (whatever the I3C acts as controller/target)

Bit 10 **RXTGTENDMIS**: target-initiated read end masked interrupt status (when the I3C acts as controller)

Bit 9 **FCMIS**: frame complete masked interrupt status (whatever the I3C acts as controller/target)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEMIS**: RX-FIFO not empty masked interrupt status (whatever the I3C acts as controller/target)

Bit 4 **TXFNFMIS**: TX-FIFO not full masked interrupt status (whatever the I3C acts as controller/target)

Bit 3 **SFNEMIS**: S-FIFO not empty masked interrupt status (when the I3C acts as controller)

Bit 2 **CFNFMIS**: C-FIFO not full masked interrupt status (when the I3C acts as controller)

Bits 1:0 Reserved, must be kept at reset value.

46.16.17 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.

This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:

– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.

– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset. This (see) has the same impact as a pin reset (NRST = 0):

– the software writes and sets the SYSRESETREQ control bit of the AITR register, when the device is controlled by a Cortex®-M.

– the software writes and sets SYSRST = 1 in the RCC_GRSTCSETR register, when the device is controlled by a Cortex®-A.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

46.16.18 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.
- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;
- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

46.16.19 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

46.16.20 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MWL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

46.16.21 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C fm+ device t_{DIG_Hmin} = 260 ns, with I^2C fm device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C fm+ device t_{DIG_Lmin} = 500 ns, with I^2C fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

46.16.22 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SDA_HD[1:0]		Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
		rw	rw						rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SDA_HD[1:0]**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD[1:0] + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD[1:0])] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

46.16.23 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bit 7 Reserved, must be kept at reset value.

Bit 6 **STALLL:** Controller clock stall enable in the address ACK/NACK phase (before the ninth bit) of a legacy I2C read/write message. This allows the I2C target to prepare the next data byte to send/receive.

0: no stall

1: stall enabled

Bit 5 **STALLS:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C write message. This allows the I2C target to prepare the next data byte to receive.

0: no stall

1: stall enabled

Bit 4 **STALLR:** Controller clock stall enable in the data ACK/NACK phase of a legacy I2C read message. This allows the I2C target to prepare the next data byte to send.

0: no stall

1: stall enabled

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

46.16.24 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable

0: I3C target (no controller capable)

1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

46.16.25 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device
Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

46.16.26 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

46.16.27 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

46.16.28 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

46.16.29 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID

This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).

This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector

This field is set as 0 (vendor fixed value).

This field represents bit[32] of the 48-bit provisioned ID.

Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID

This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.

This field represents bits[15:12] of the 48-bit provisioned ID.

Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

46.16.30 I3C register map

Table 490. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	TRIGHWEN	TRIGPOL	TRIGSEL [3:0]				Res.	FCFDIS	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]																
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_m_TX	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 490. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	I3C_MISR	GRPMIS	DEFMIS	INTUPDMIS	ASUPDMIS	RSTMIS	MRLUPDMIS	MWLUPDMIS	DAUPDMIS	STAMIS	GETMIS	WKPMIS	Res.	HJMIS	CRUPDMIS	CRMIS	IBIENDMIS	IBIMIS	Res.	Res.	Res.	ERRMIS	RXTGTENDMIS	FCMIS	Res.	Res.	Res.	RXFNEMIS	TXFNFMIS	SFNEMIS	CFNFMIS	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0				0	0	0	0		
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							TS
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	SDA_HD		Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value			0	0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	STALLL	STALLS	STALLR	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 490. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.3* for the register boundary addresses.

47 Universal synchronous/asynchronous receiver transmitter (USART/UART)

47.1 Introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

47.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  – Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

47.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

47.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 491. Instance implementation on STM32U3 series

Instance	STM32U375/385
USART1	Full
USART3	Full
UART4	Basic
UART5	Basic
LPUART1	Low-power

Table 492. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-

Table 492. USART/LPUART features (continued)

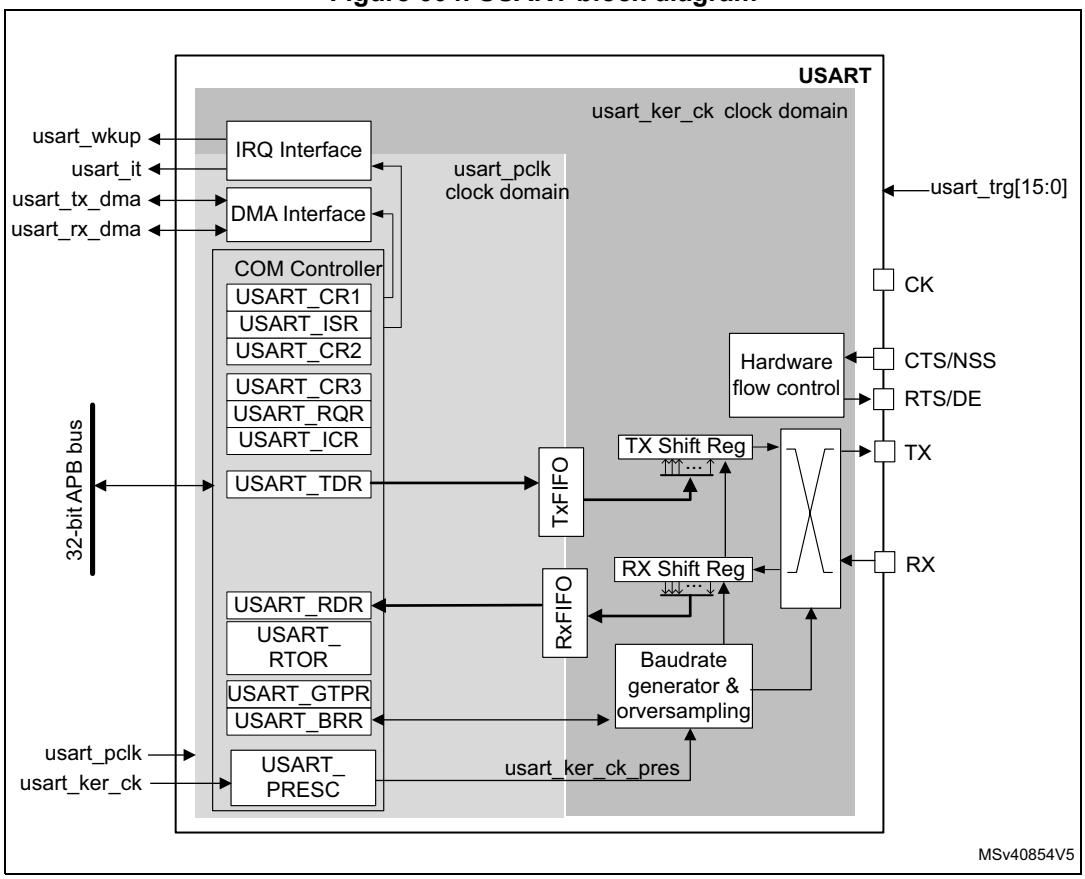
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[3]
Autonomous mode	X	X	X

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop 0 and Stop 1 modes.
3. Wake-up supported from Stop 0, Stop 1 and Stop 2 modes.

47.5 USART functional description

47.5.1 USART block diagram

Figure 604. USART block diagram



47.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 493* and *Table 494* for the list of USART input/output pins and internal signals.

Table 493. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 494. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request
usart_trg[15:0]	Input	USART triggers.

Description of USART interconnections

Table 495. USART interconnection (USART1/3 and UART4/5)

Signal name	Source
usart_trg0	gpdma1_ch0_tcf
usart_trg1	gpdma1_ch1_tcf
usart_trg2	gpdma1_ch2_tcf
usart_trg3	gpdma1_ch3_tcf
usart_trg4	exti6
usart_trg5	exti9
usart_trg6	lptim1_out
usart_trg7	lptim2_out
usart_trg8	comp1_out
usart_trg9	comp2_out
usart_trg10	rtc_alra_trg
usart_trg11	rtc_wut_trg
usart_trg12	-
usart_trg13	-
usart_trg14	-
usart_trg15	-

47.5.3 USART clocks

The simplified block diagram given in *Section 47.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

47.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 605*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

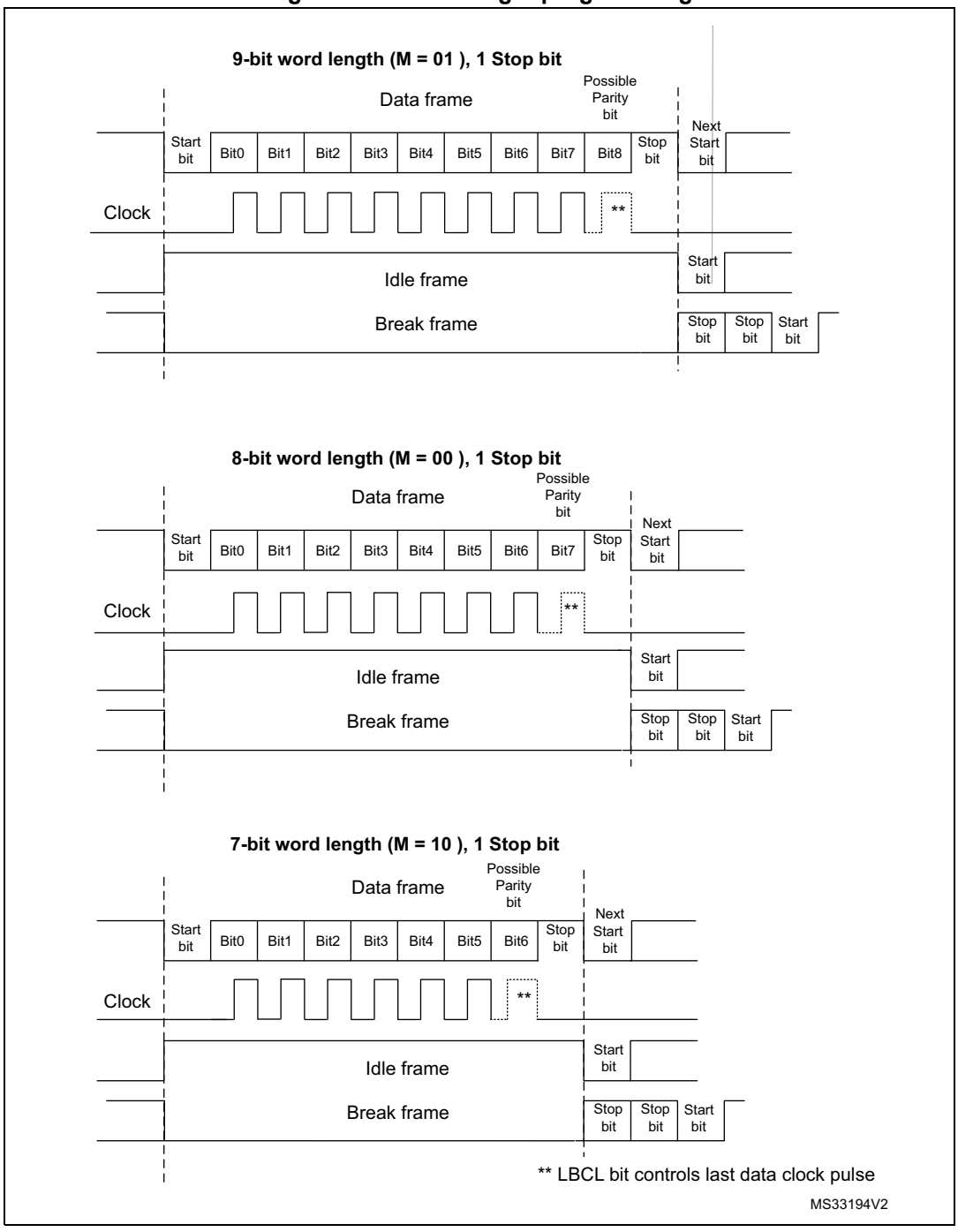
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 605. Word length programming



47.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

47.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

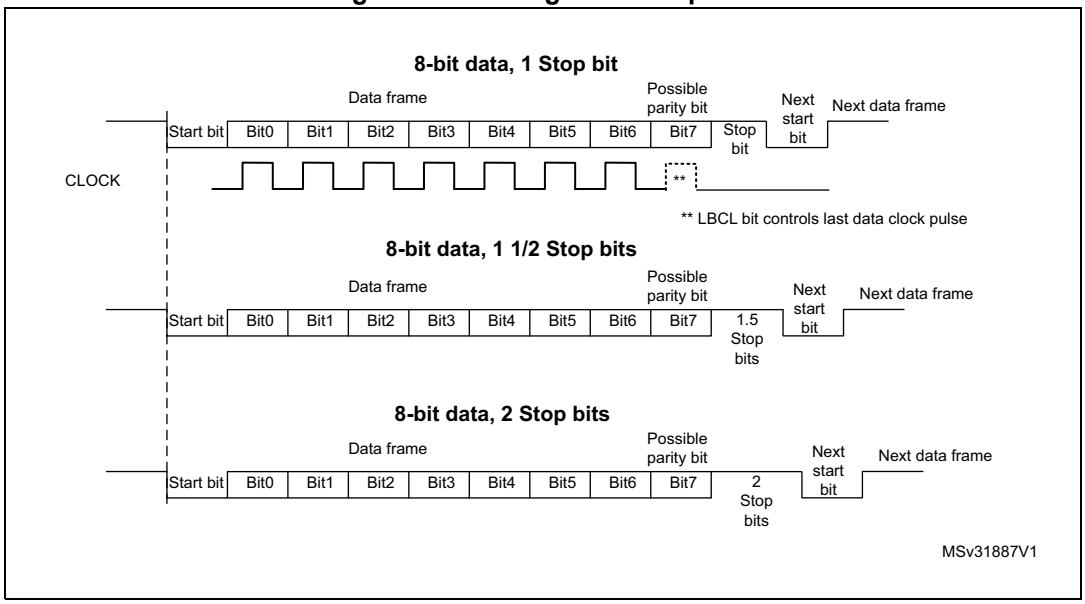
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 47.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 606. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 47.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

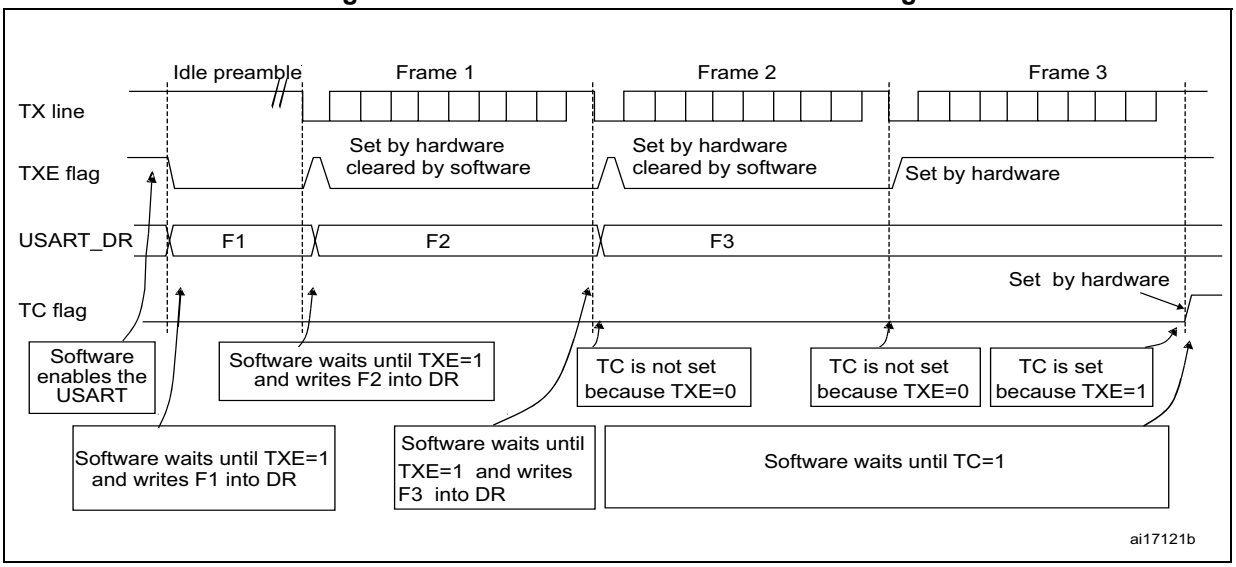
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 607*).

Figure 607. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 605*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

47.5.7 USART receiver

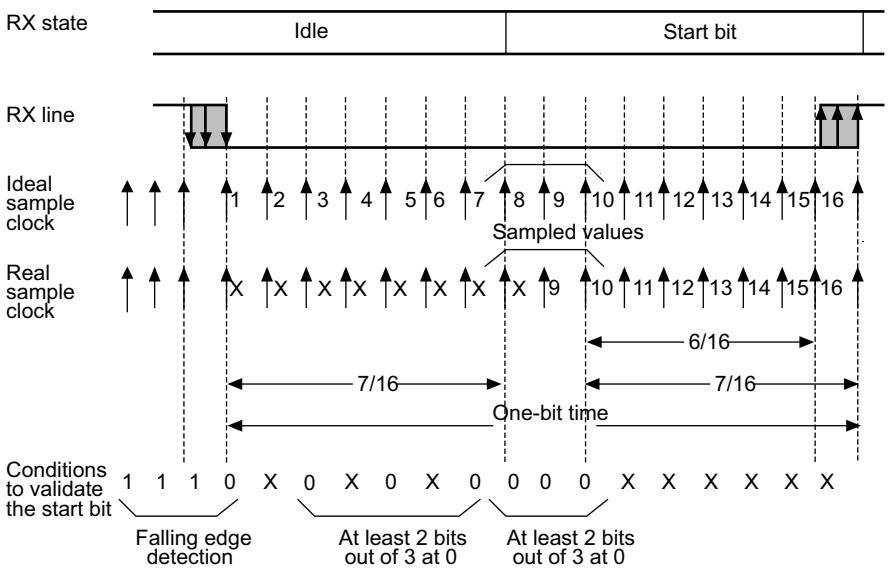
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 608. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 47.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

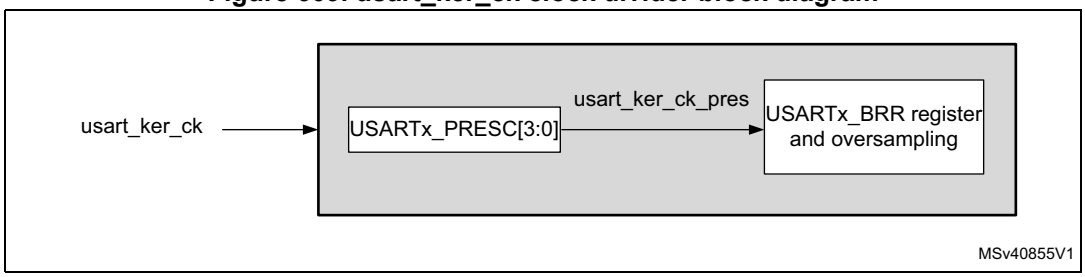
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 609. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 610* and *Figure 611*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 47.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 496*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 47.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 610. Data sampling when oversampling by 16

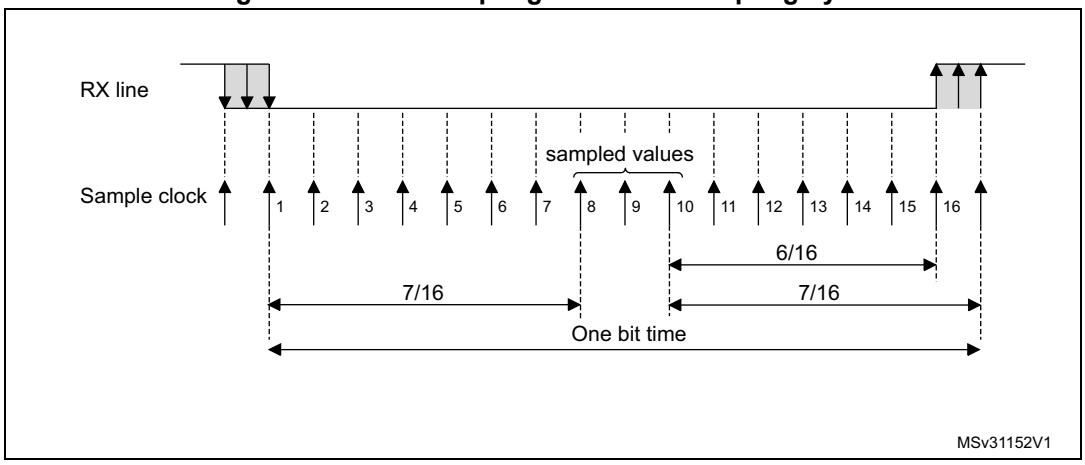


Figure 611. Data sampling when oversampling by 8

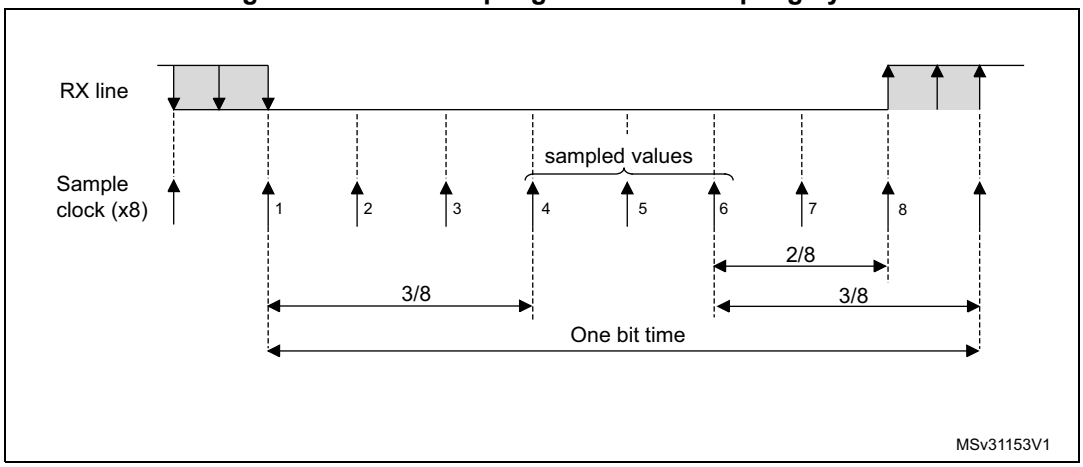


Table 496. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 47.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

47.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

47.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 497*, *Table 498*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 497. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 498. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 497 and Table 498 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

47.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

47.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

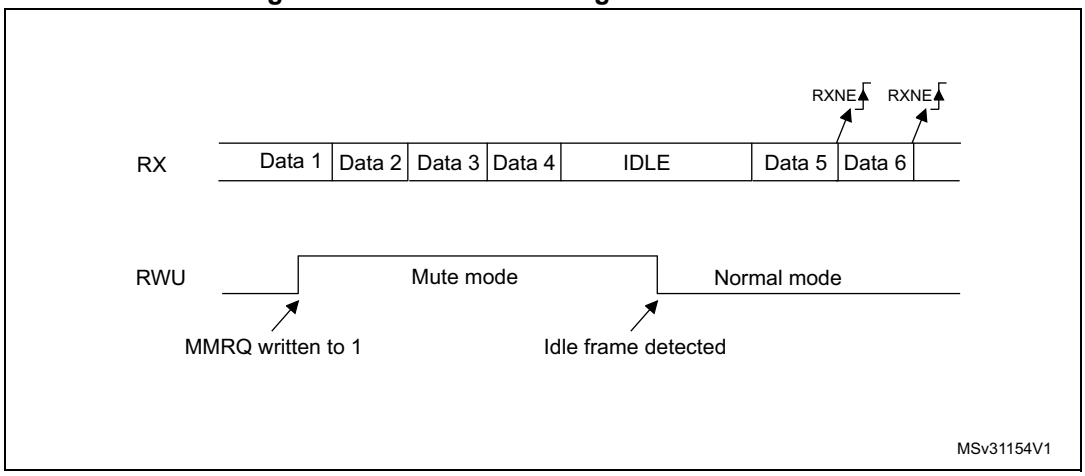
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 612*.

Figure 612. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

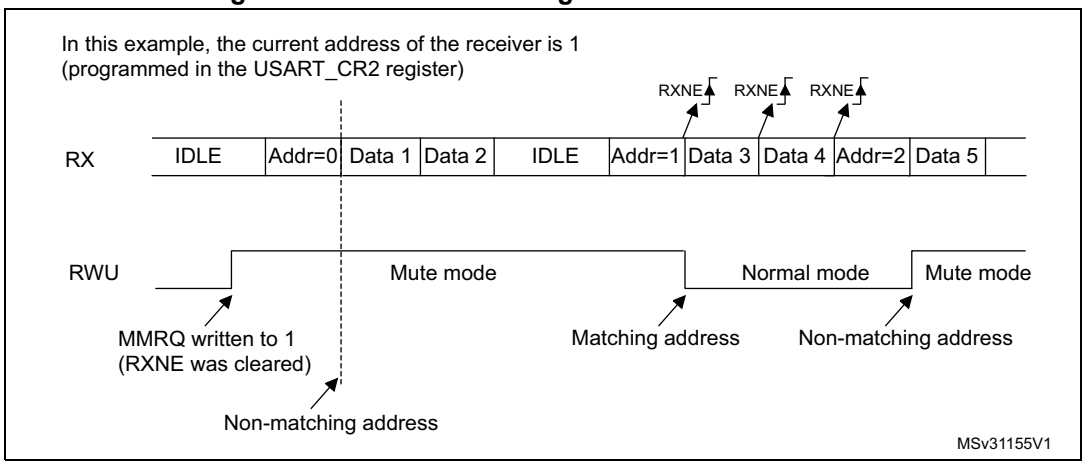
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 613*.

Figure 613. Mute mode using address mark detection



47.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

47.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 499*.

Table 499. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

47.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 47.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 47.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 614*.

Examples of break frames are given in *Figure 615*.

Figure 614. Break detection in LIN mode (11-bit break length - LBDL bit is set)

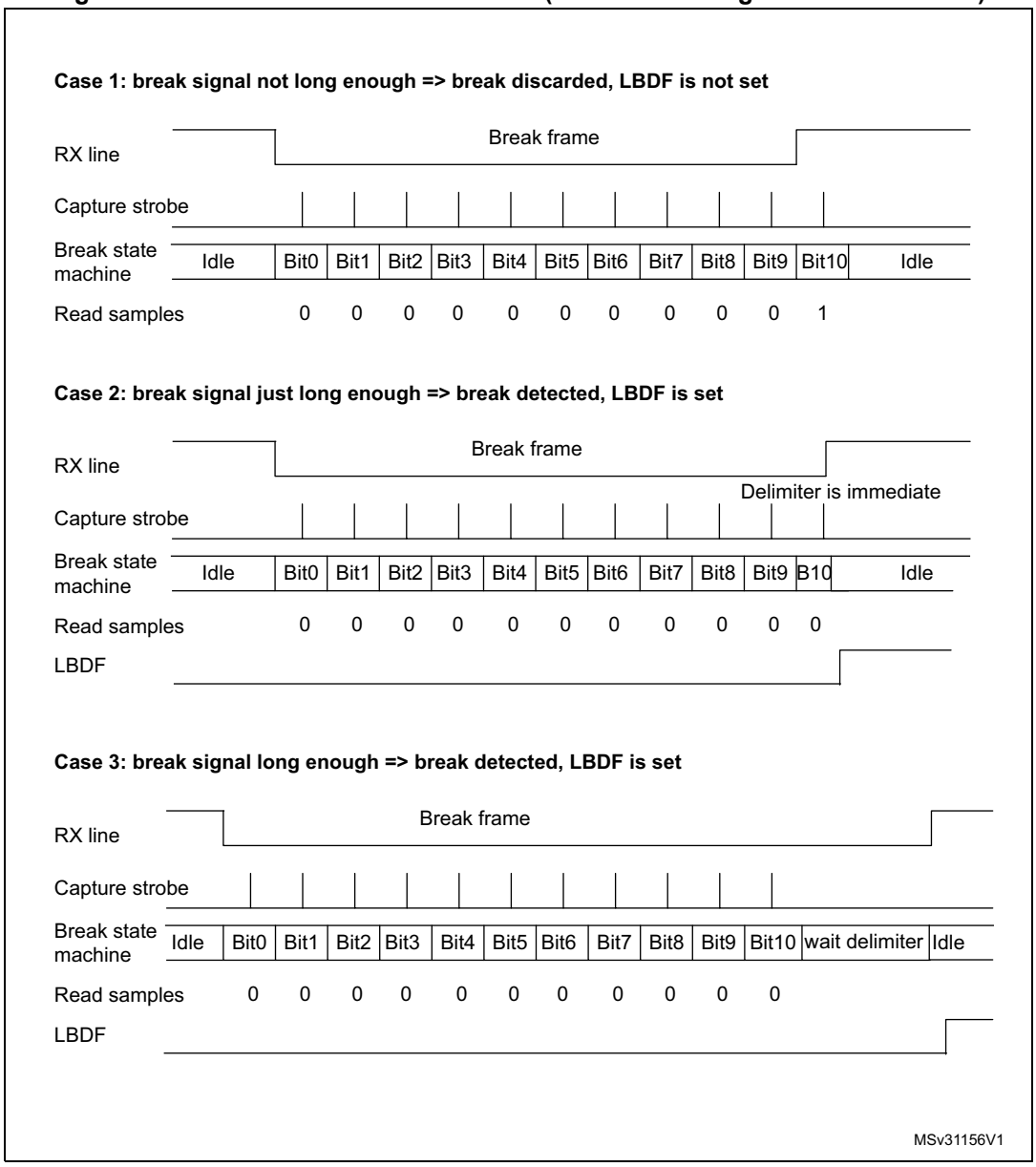
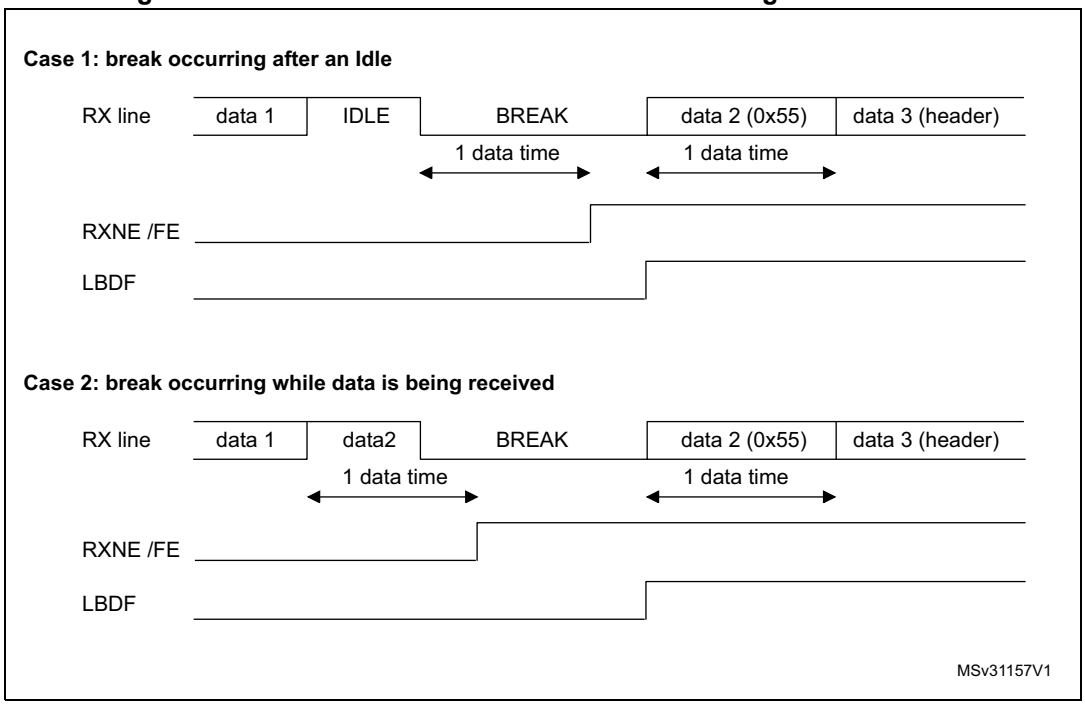


Figure 615. Break detection in LIN mode vs. Framing error detection



47.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 616*, *Figure 617* and *Figure 618*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 616. USART example of synchronous master transmission

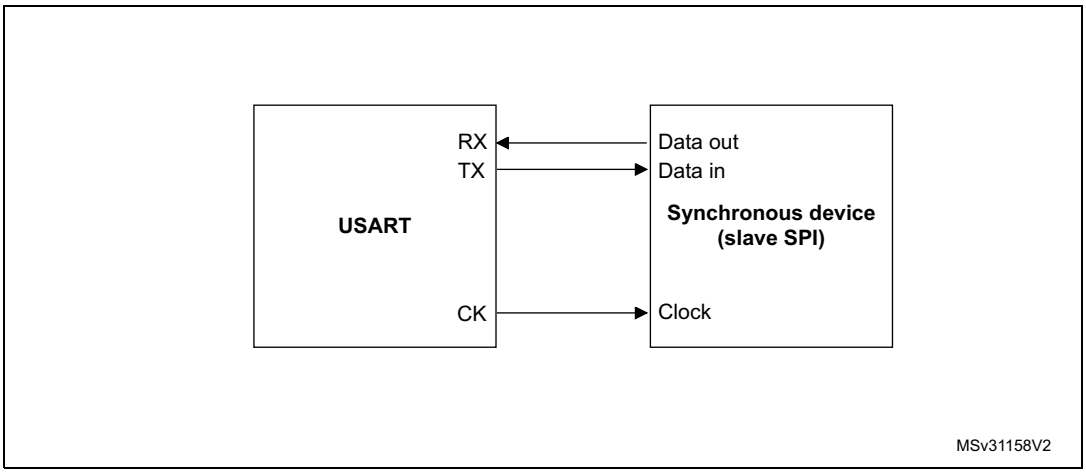


Figure 617. USART data clock timing diagram in synchronous master mode (M bits = 00)

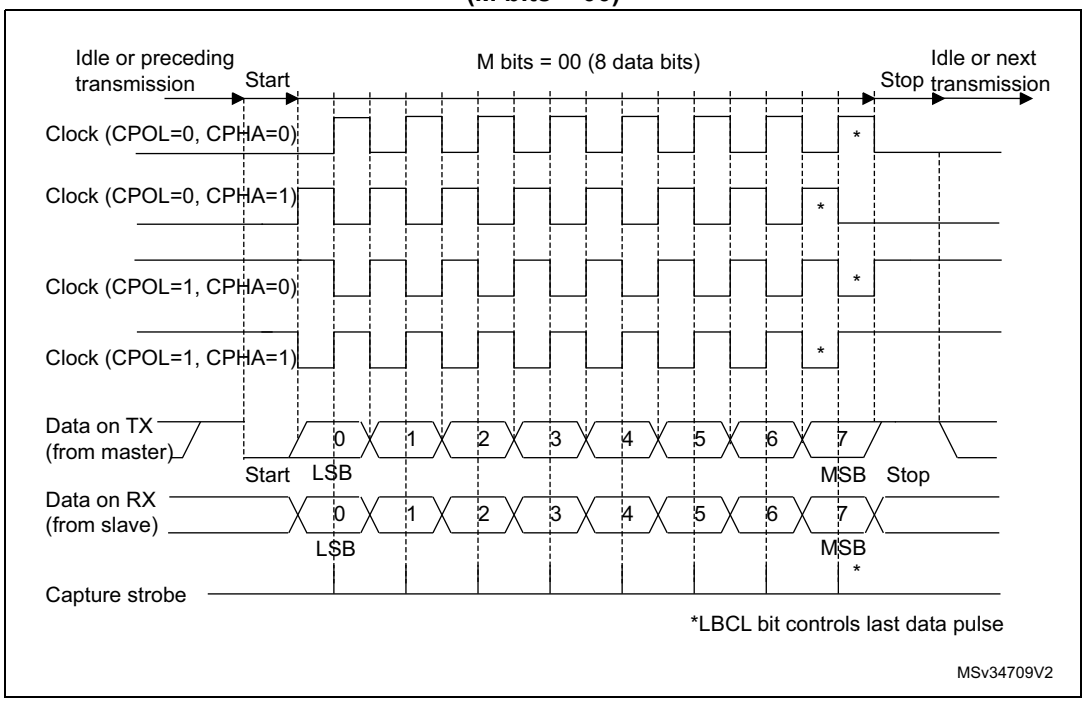
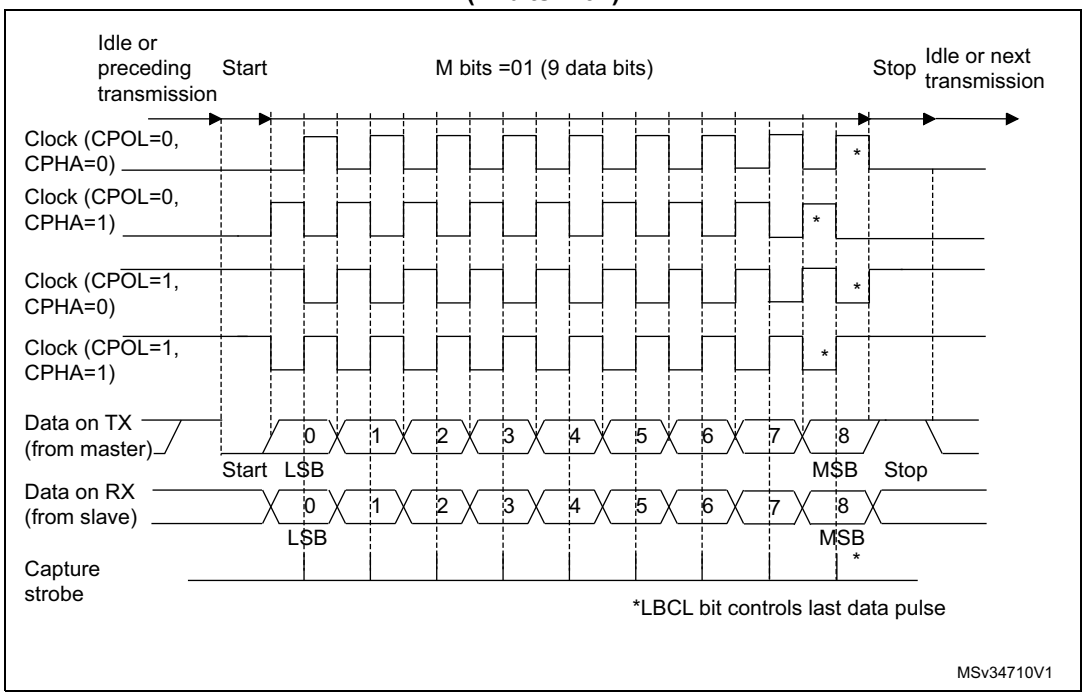


Figure 618. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

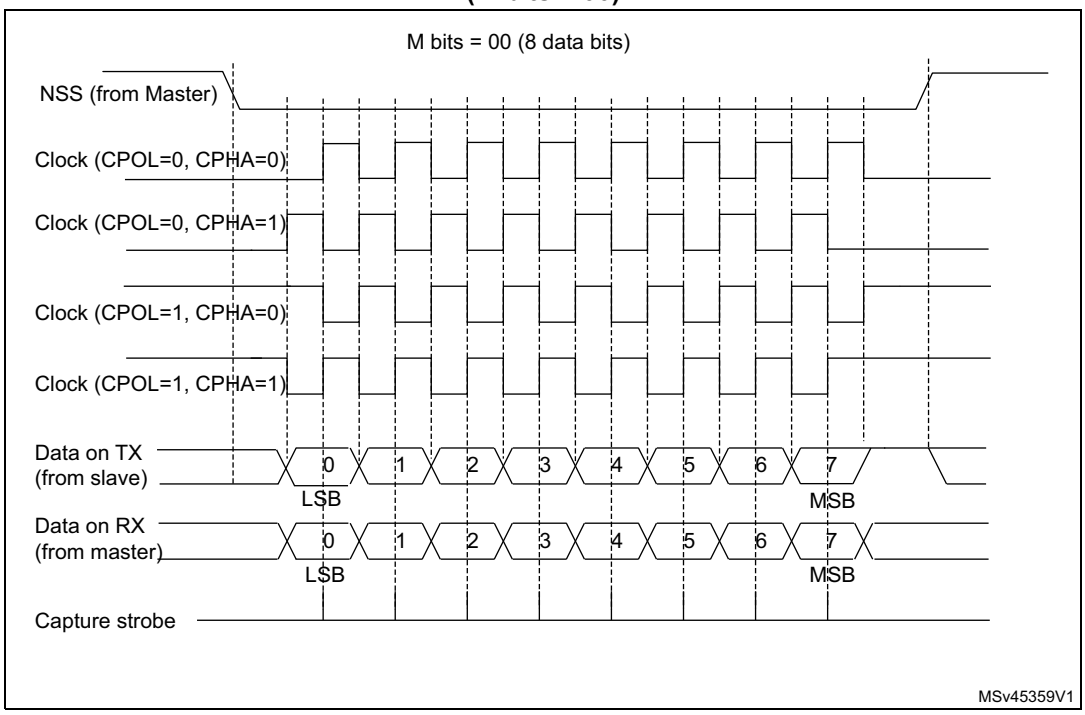
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 619*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 619. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

47.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

47.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

47.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 47.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

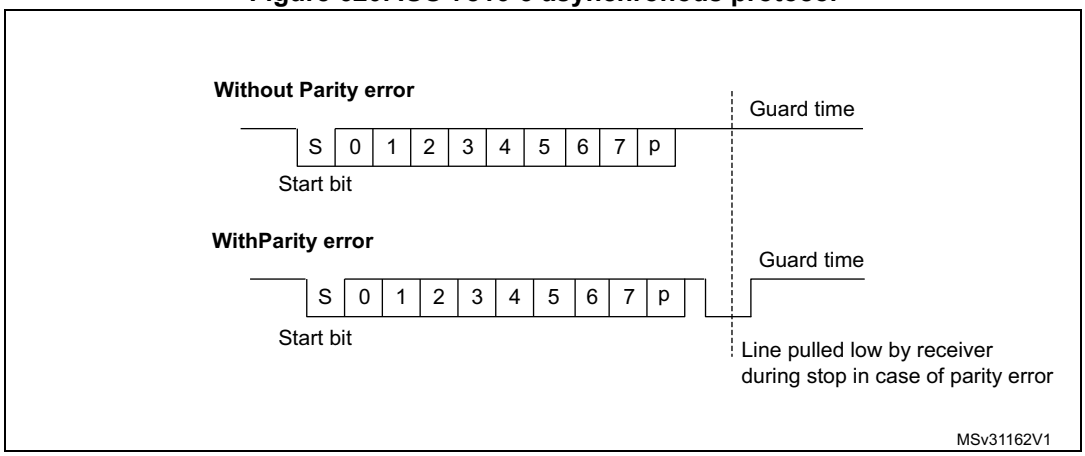
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 620 shows examples of what can be seen on the data line with and without parity error.

Figure 620. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

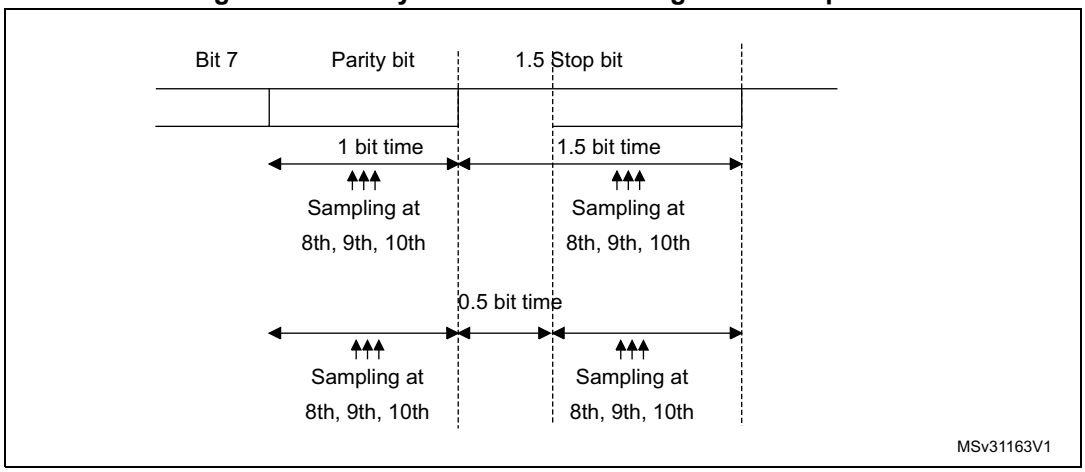
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 621 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 621. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

47.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 47.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 622*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 623*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 622. IrDA SIR ENDEC block diagram

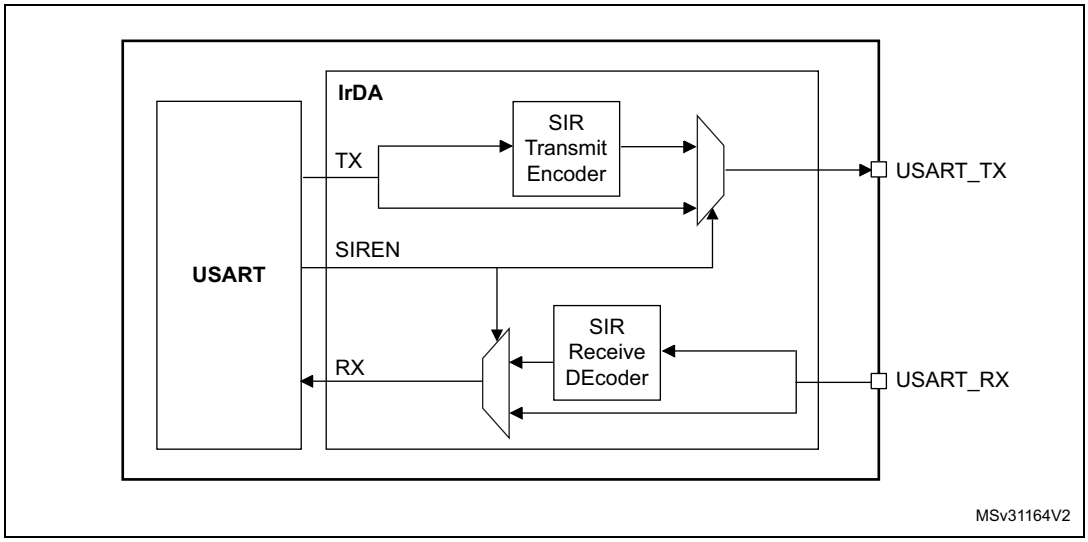
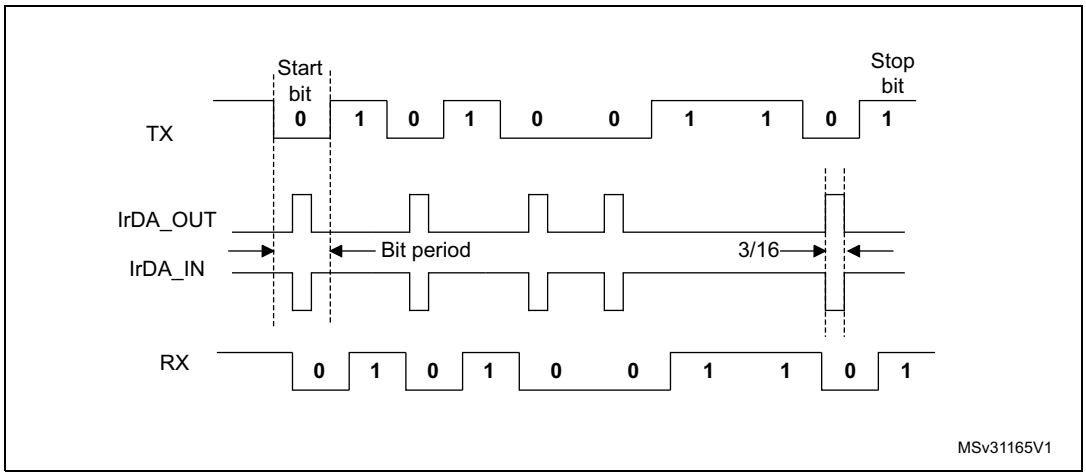


Figure 623. IrDA data modulation (3/16) - normal mode



47.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 47.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 47.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

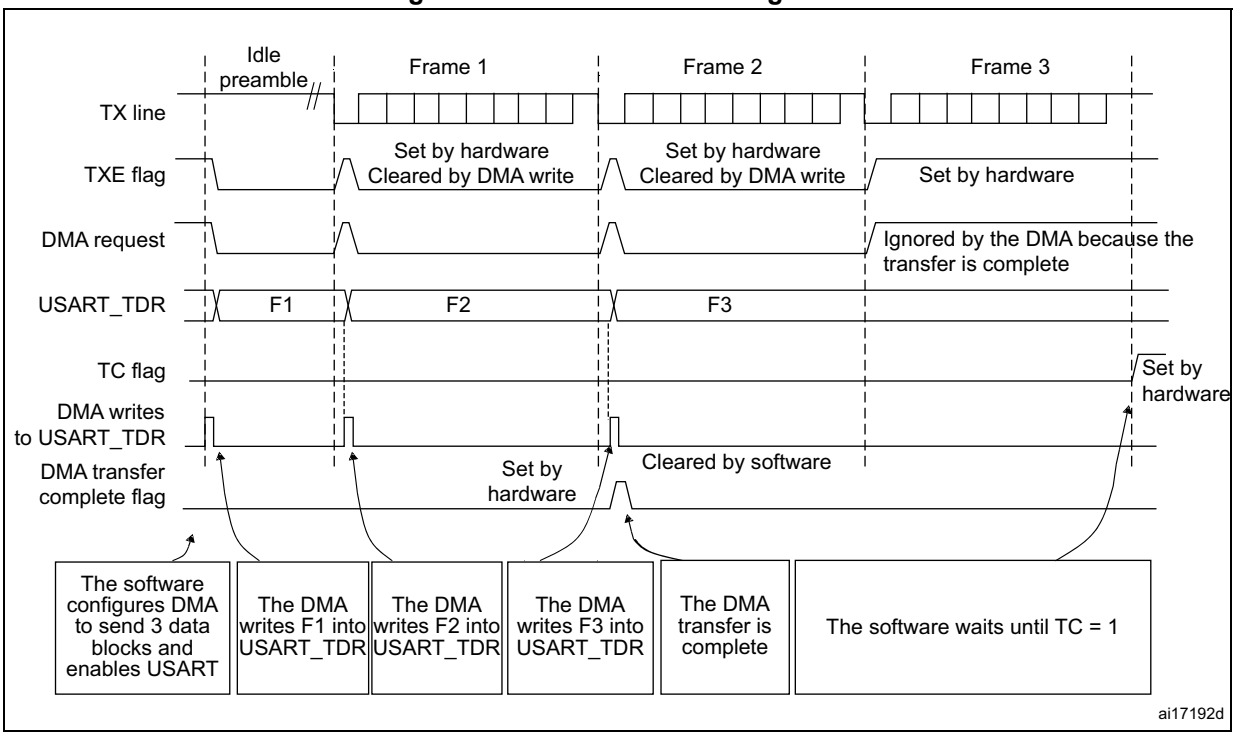
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 624. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

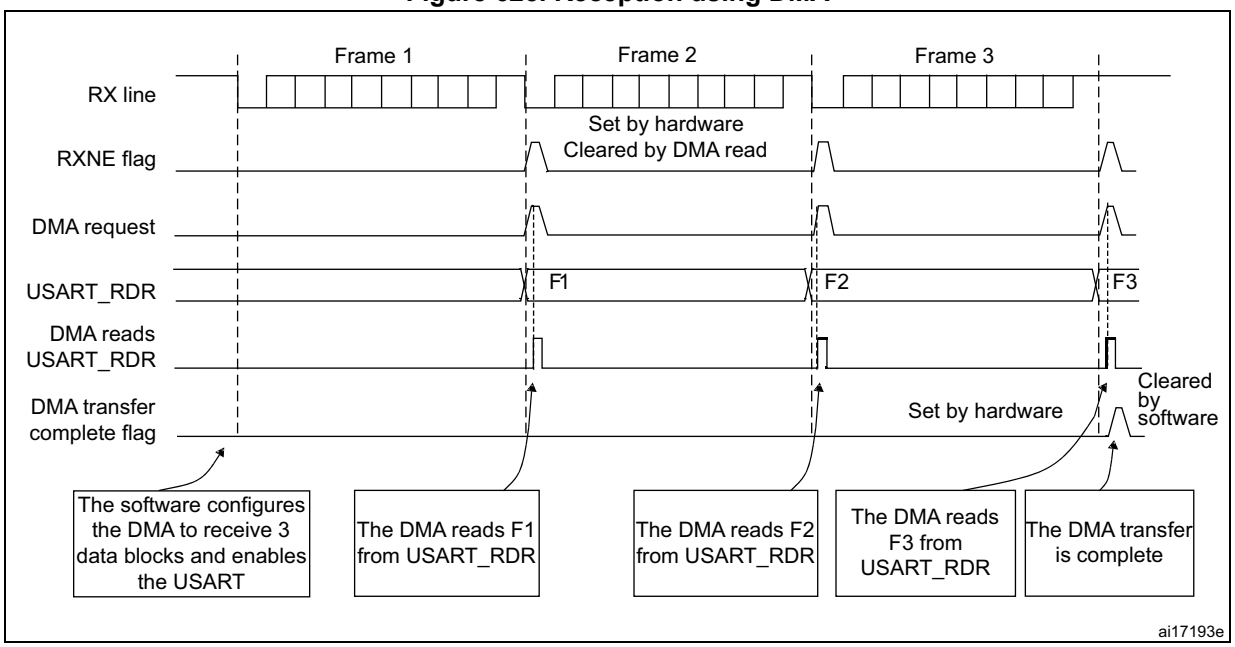
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 625. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

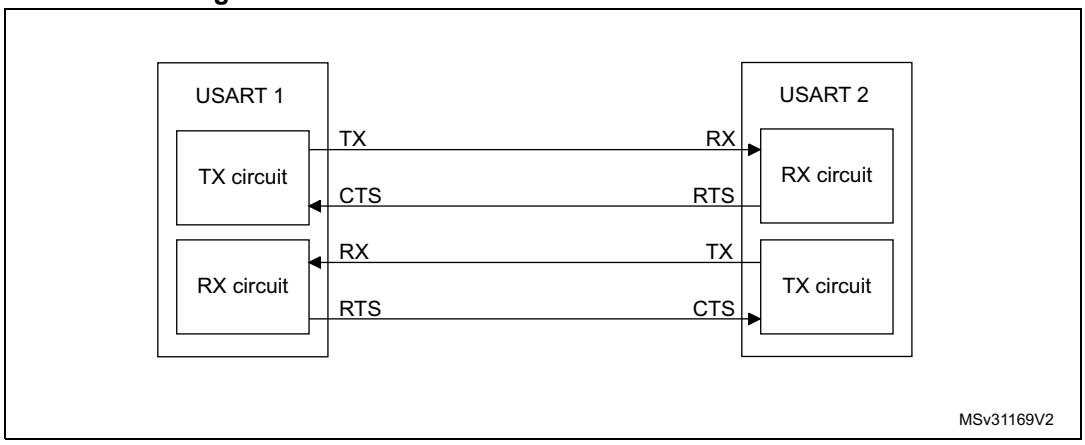
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

47.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 626* shows how to connect two devices in this mode:

Figure 626. Hardware flow control between two USARTs

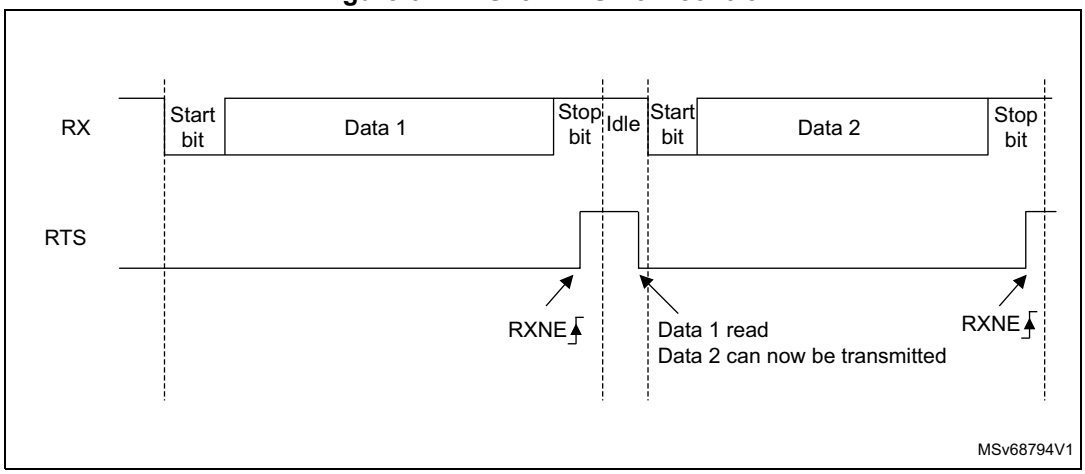


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 627* shows an example of communication with RTS flow control enabled.

Figure 627. RS232 RTS flow control



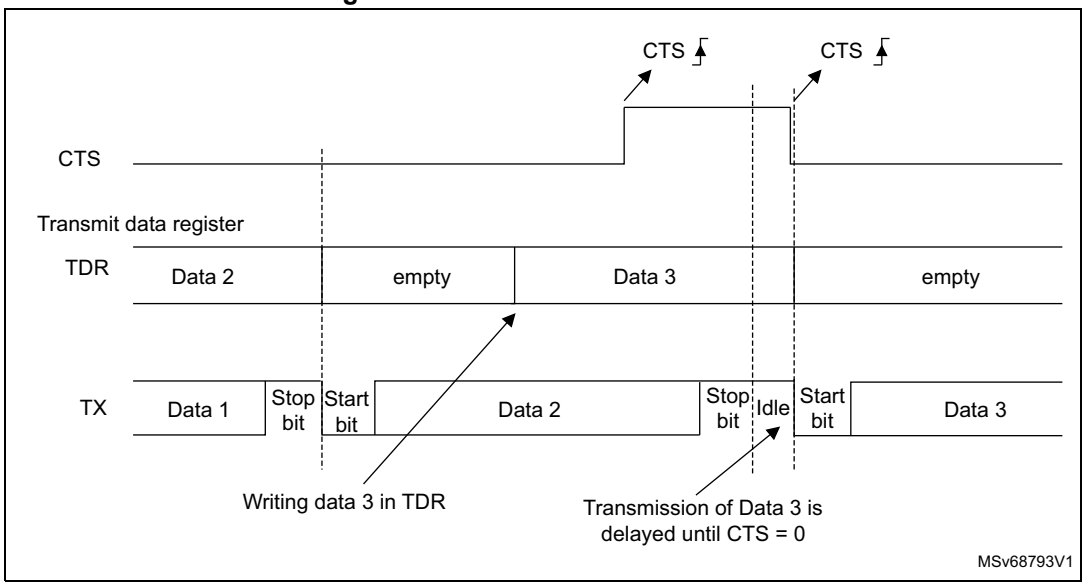
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 628* shows an example of communication with CTS flow control enabled.

Figure 628. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

47.5.22 USART autonomous mode

The USART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the USART status needs to be updated. Once the USART receives the APB clock, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the kernel and APB clocks are still available for the USART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

USART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the USART_AUTOCR register. It sets the TE bit in the USART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the USART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

USART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1 or DMAR = 1)
- If the FIFO mode is disabled, the APB clock is requested when the USART finishes sampling data and it is ready to be written in the USART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation of an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

The APB clock is also requested in reception mode when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Only UART and SPI master modes support the autonomous mode.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 47.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 497: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

47.6 USART in low-power modes

Table 500. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. If the USART is clocked by an oscillator available in Stop mode, transfers in asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 47.4* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

47.7 USART interrupts

Refer to *Table 501* for a detailed description of all USART interrupt requests.

Table 501. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 501. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(4)	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(4)	
	Overrun error in multibuffer communication.	ORE(5)		Write 1 in ORECF		Yes	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		Yes(4)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(6)	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 47.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
5. When OVRDIS = 0.
6. The DMA must be used when the FIFO mode is enabled.

47.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

47.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

47.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits

M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

47.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

– In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.

– In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.

– In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 610* and *Figure 611*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length
This bit is for selection between 11 bit or 10 bit break detection.
0: 10-bit break detection
1: 11-bit break detection
This bit can only be written when the USART is disabled (UE = 0).
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the USART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:
When the DIS_NSS bit is set, the NSS pin input is ignored.
0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

47.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	Res.	Res.	Res.	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation on page 1974.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation on page 1974.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

47.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCARCNT[2:0]			Res.
							rw					rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 47.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

47.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

47.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 47.4: USART implementation.

47.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 47.4: USART implementation.

47.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

47.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 47.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by writing to the USART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 47.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

47.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	Res.	RWU	SBKF	CMF	BUSY
						r			r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 47.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 47.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

47.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

47.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 47.5.1: USART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

47.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 47.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

47.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 47.4: USART implementation.

47.8.16 USART autonomous mode control register (USART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description of USART interconnections*.
This bitfield can be written only when the UE bit is cleared in USART_CR1 register.
0000: usart_trg0 selected
0001: usart_trg1 selected
...
1111: usart_trg15 selected

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in USART_CR1)
1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in USART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in USART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDN transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in USART_CR1.

47.8.17 USART register map

Table 502. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	Res.	Res.	Res.	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0					0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 502. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0
0x30	USART_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]															
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

48 Low-power universal asynchronous receiver transmitter (LPUART)

48.1 Introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

48.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

48.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 503. Instance implementation on STM32U3 series

Instance	STM32U375/385
USART1	Full
USART3	Full
UART4	Basic
UART5	Basic
LPUART1	Low-power

Table 504. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-

Table 504. USART/LPUART features (continued)

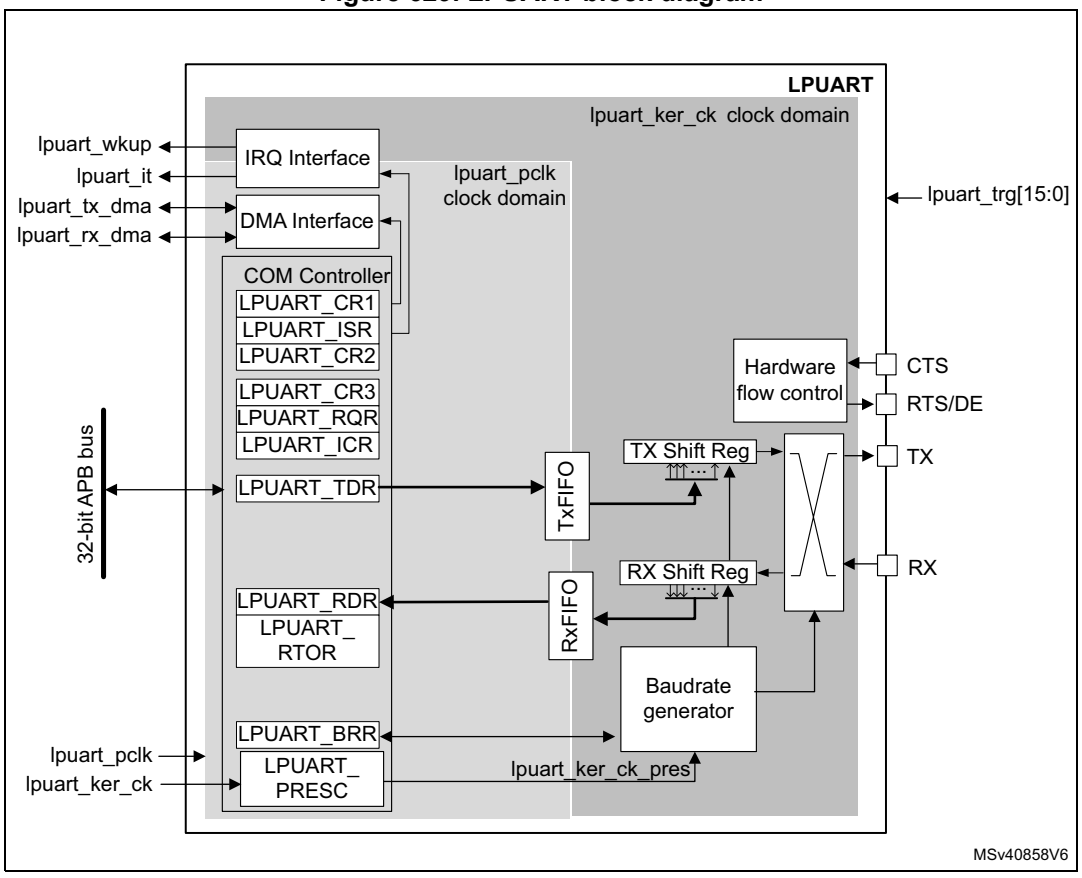
Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X(2)	X(2)	X(3)
Autonomous mode	X	X	X

1. X = supported.
2. Wake-up supported from Stop 0 and Stop1 modes.
3. Wake-up supported from Stop 0, Stop 1 and Stop 2 modes.

48.4 LPUART functional description

48.4.1 LPUART block diagram

Figure 629. LPUART block diagram



48.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 505* and *Table 506* for the list of LPUART input/output pins and internal signals.

Table 505. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 506. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 506. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request
lpuart_trg[15:0]	Input	LPUART triggers

Description LPUART interconnections

Table 507. LPUART interconnections (LPUART1)

Signal name	Source
lpuart_trg0	gpdma1_ch0_tcf
lpuart_trg1	gpdma1_ch1_tcf
lpuart_trg2	gpdma1_ch2_tcf
lpuart_trg3	gpdma1_ch3_tcf
lpuart_trg4	exti6
lpuart_trg5	exti8
lpuart_trg6	lptim1_out
lpuart_trg7	lptim3_out
lpuart_trg8	comp1_out
lpuart_trg9	comp2_out
lpuart_trg10	rtc_alra_trg
lpuart_trg11	rtc_wut_trg
lpuart_trg12	-
lpuart_trg13	-
lpuart_trg14	-
lpuart_trg15	-

48.4.3 LPUART clocks

The simplified block diagram given in *Section 48.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

48.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 605*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

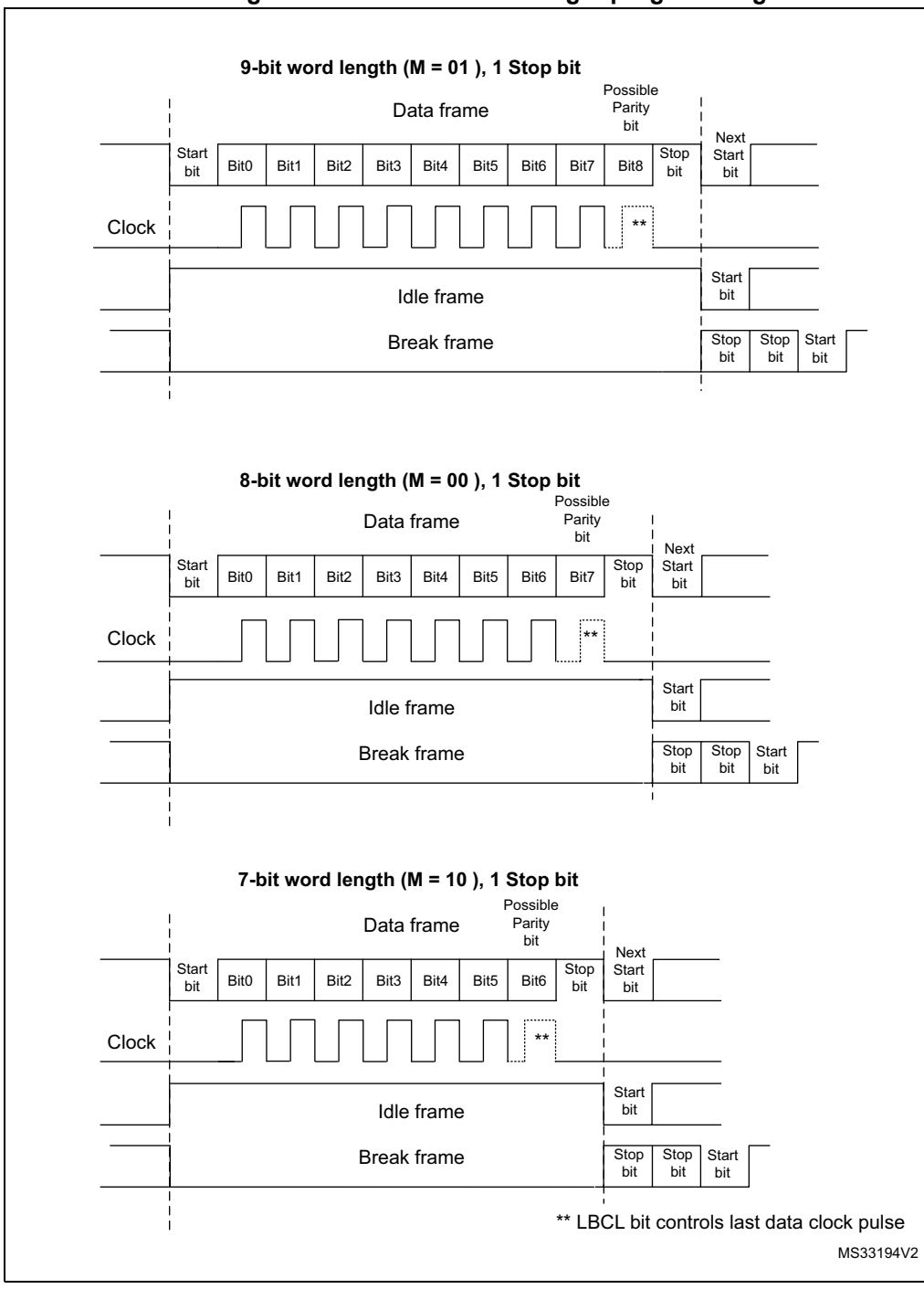
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 630. LPUART word length programming



48.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

48.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 48.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

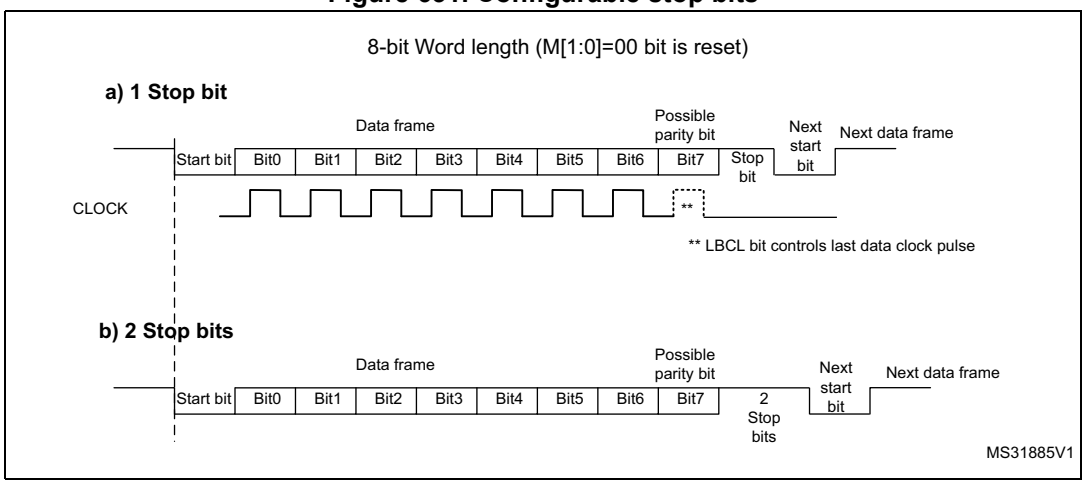
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 631. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 48.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

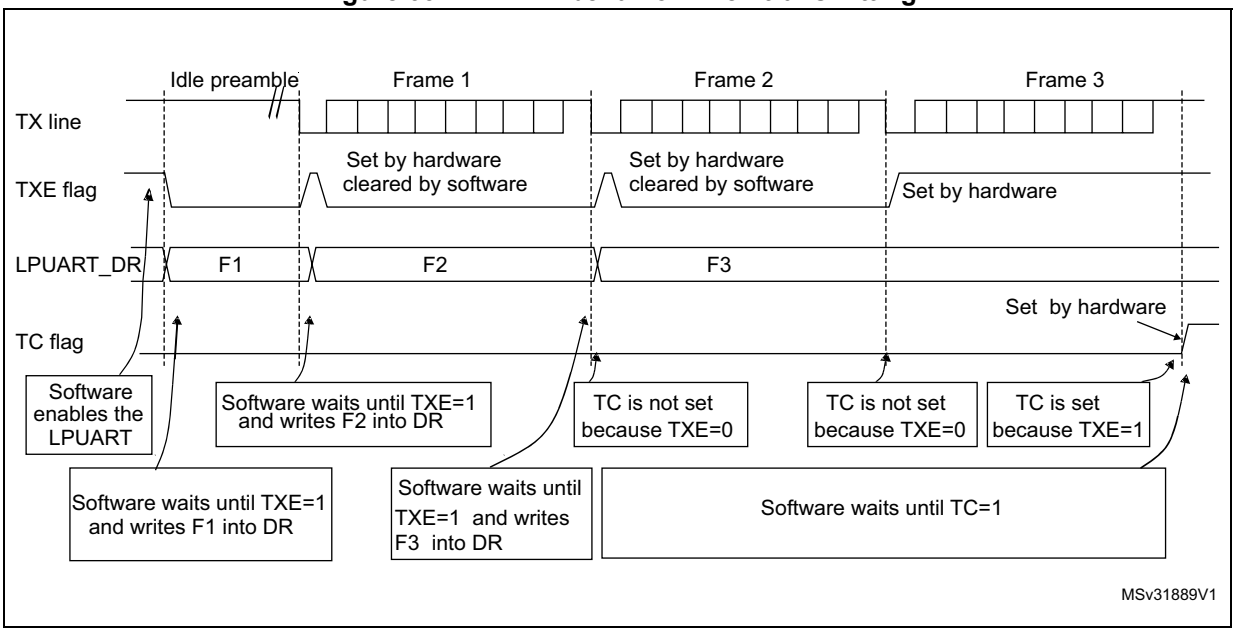
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 632: TC/TXE behavior when transmitting*).

Figure 632. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 630*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

48.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 48.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

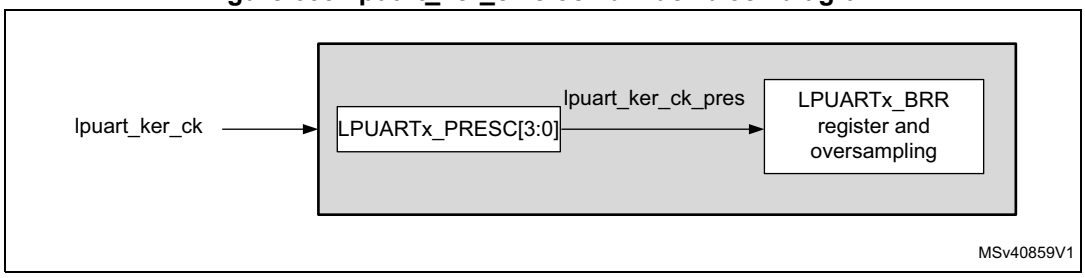
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 633. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

48.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 508. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

48.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 509*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 509. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 509 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

48.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

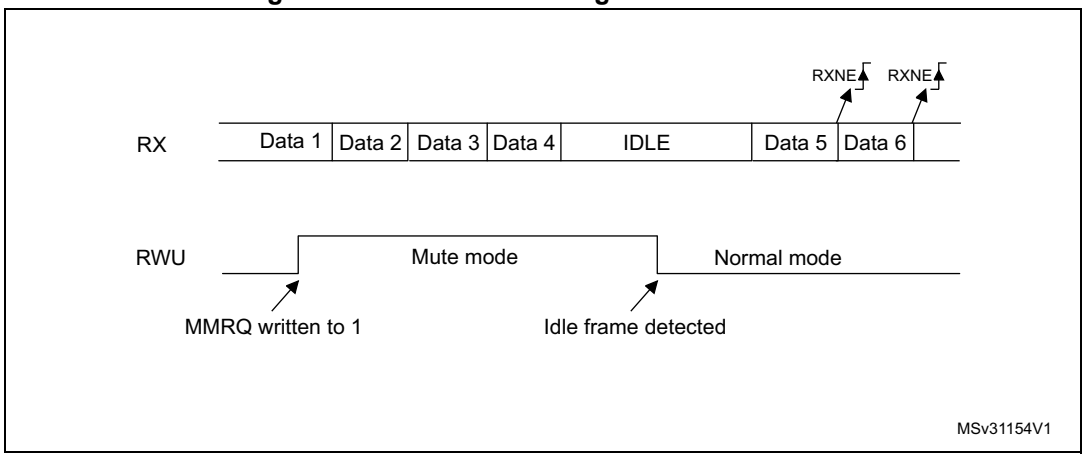
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 634*.

Figure 634. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

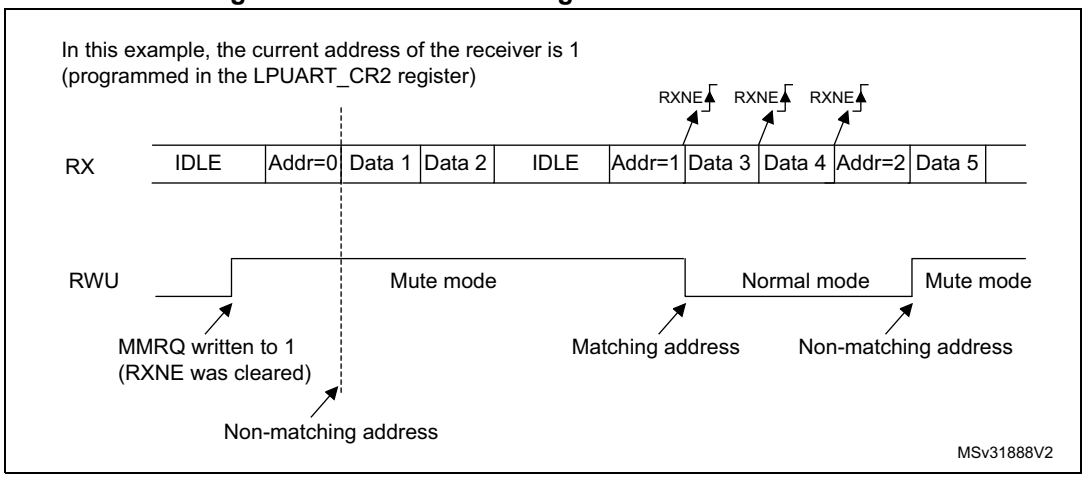
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 635*.

Figure 635. Mute mode using address mark detection



48.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 510*.

Table 510: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

48.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

48.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 48.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 48.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

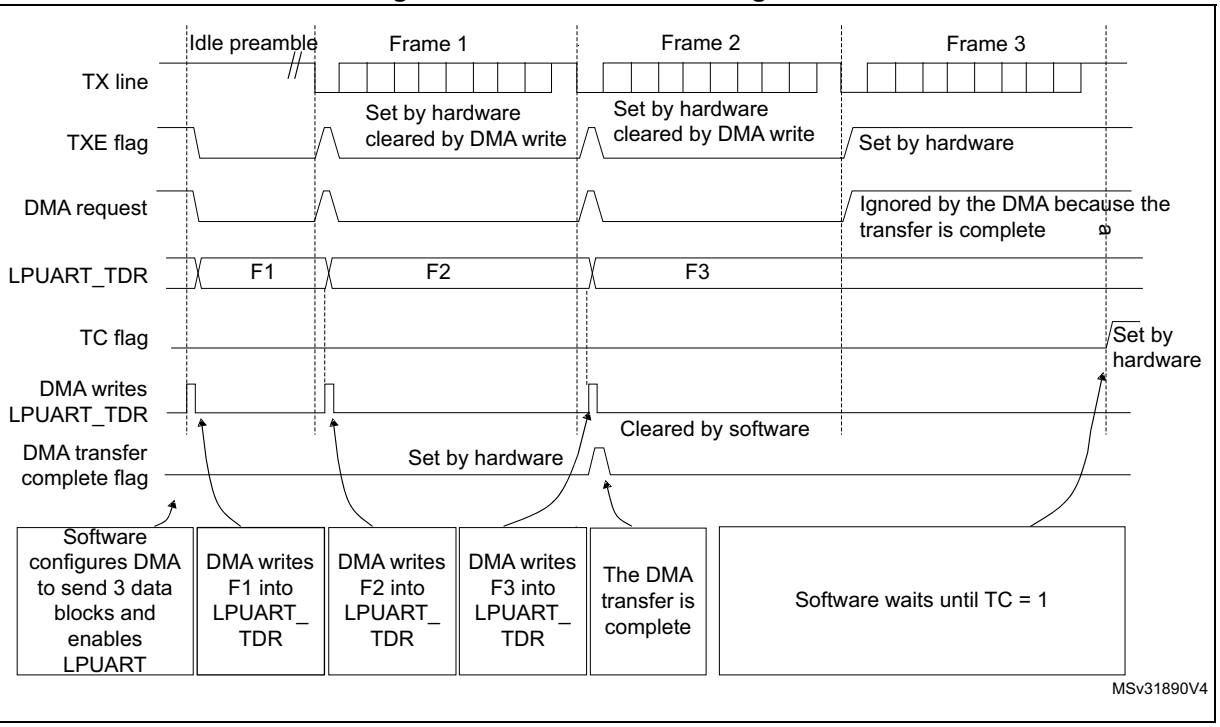
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 636. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

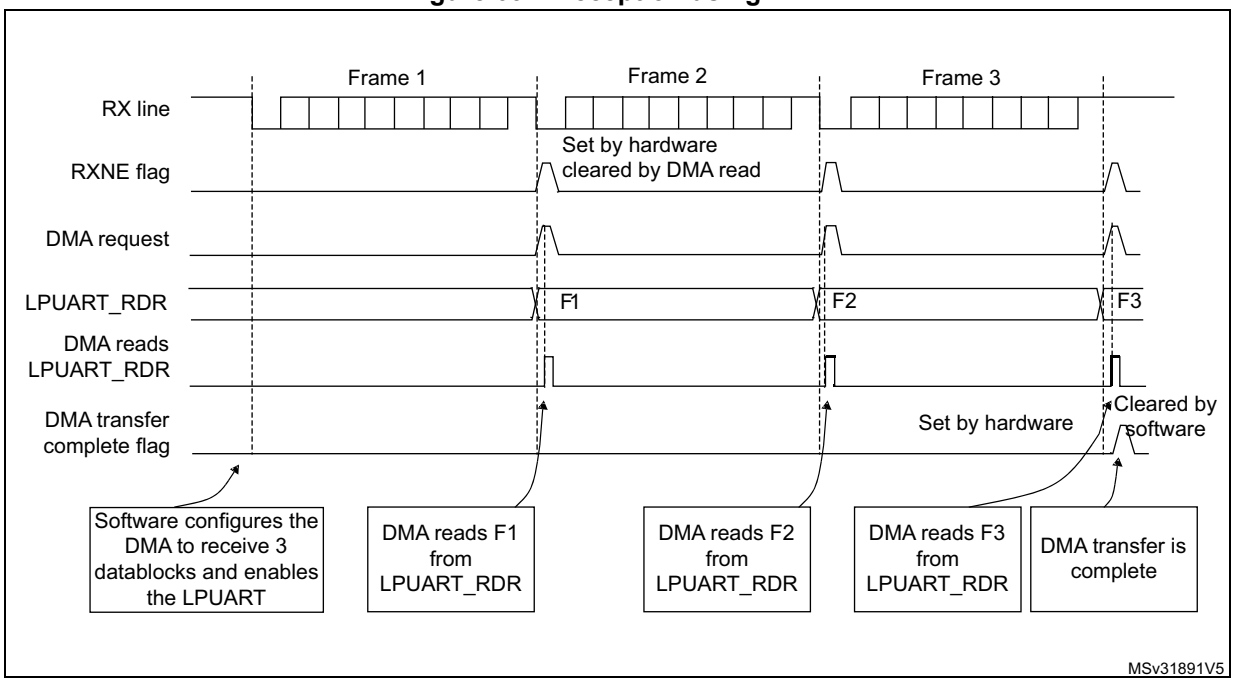
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 637. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

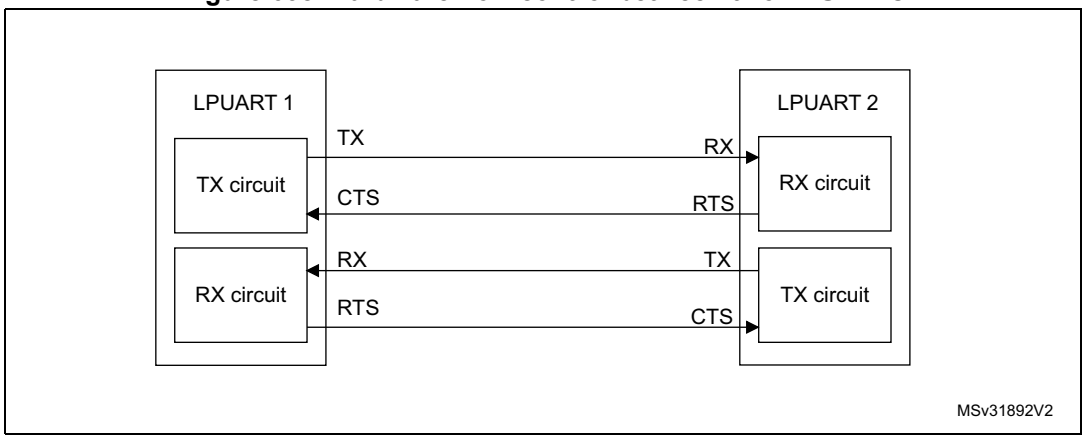
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

48.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 638* shows how to connect two devices in this mode.

Figure 638. Hardware flow control between two LPUARTs

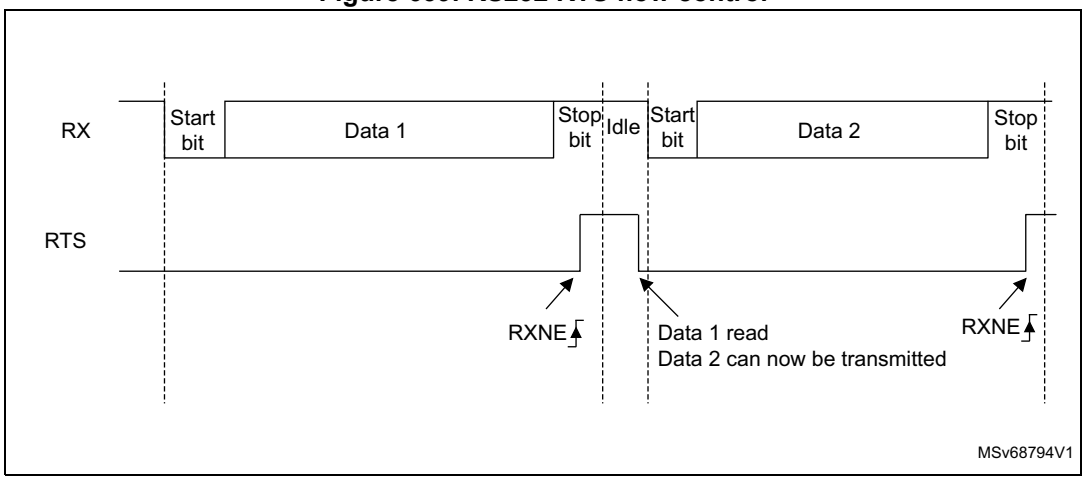


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 639* shows an example of communication with RTS flow control enabled.

Figure 639. RS232 RTS flow control



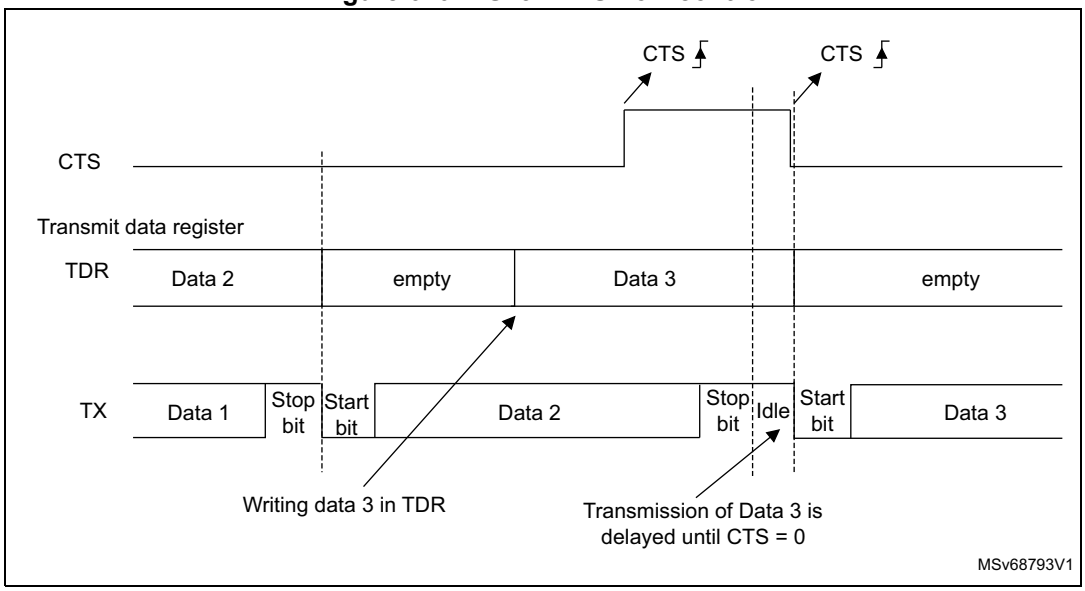
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 640* shows an example of communication with CTS flow control enabled.

Figure 640. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

48.4.15 LPUART autonomous mode

The LPUART peripheral can be functional in Stop mode thanks to the autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the LPUART status needs to be updated. Once the LPUART receives the kernel and APB clocks, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the APB clock is still available for the LPUART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

LPUART transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - The TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - The TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - The TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep, or Stop mode. The trigger is selected through the TRIGSEL bit in the LPUART_AUTOCR register. It sets the TE bit in the LPUART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bitfield in the LPUART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

LPUART reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - The RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - The RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - The RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1)
- If the FIFO mode is disabled, the APB clock is requested when the LPUART finishes sampling data and it is ready to be written in the LPUART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

In reception mode, the APB clock is requested when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 48.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 509: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

48.5 LPUART in low-power modes

Table 511. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. If the LPUART is clocked by an oscillator available in Stop mode, transfers in asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 48.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

48.6 LPUART interrupts

Refer to for a detailed description of all LPUART interrupt requests.

Table 512. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	Yes	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		Yes	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	.Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(3)	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(3)	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		Yes	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		Yes(3)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(5)	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 48.3: LPUART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
4. When OVRDIS = 0.
5. The DMA must be used when the FIFO mode is enabled.

48.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

48.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 47.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 48.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is not used, TE bit is set and cleared by software. When the autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not functional in low-power mode.
1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.
If the LPUART *does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 48.3: LPUART implementation.*

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

48.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 47.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 48.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. When the autonomous mode is disabled, TE bit is set and cleared by software. When the autonomous mode is enabled, TE bit becomes a status bit, which is set and cleared by hardware.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register. In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART does not support the wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 48.3: LPUART implementation.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

48.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

48.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

48.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

48.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

48.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

48.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data has completed and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by writing to the LPUART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.
An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.
1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.
0: No noise is detected
1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

48.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY
									r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

48.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.
														w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

48.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 48.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

48.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 48.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

48.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 48.3: LPUART implementation.

48.7.14 LPUART autonomous mode control register (LPUART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]				IDLEDIS	TRIGEN	TRIGPOL
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description LPUART interconnections*.

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0000: lpuart_trg0 selected

0001: lpuart_trg1 selected

...

1111: lpuart_trg15 selected

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in LPUART_CR1)

1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in LPUART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDC transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in LPUART_CR1.

48.7.15 LPUART register map

Table 513. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]					Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 513. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3FIFO mode enabled	TXFTCFG[2:0]		RXFTIE		RXFTCFG[2:0]			Res.	TXFTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0								0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
	Reset value																	0	0	0	0		0	0	0	0	0			0			0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value															0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Table 513. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	LPUART_ AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL [3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[15:0]																
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

49 Serial peripheral interface (SPI)

49.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

49.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

- Autonomous functionality in Stop modes (handling of the transfer flow and required clock distribution) with wake-up from Stop capability
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

49.3 SPI implementation

The table below describes the SPI implementation.

Table 514. SPI features

SPI feature	SPI1, SPI2 (full-featured instances)	SPI3 (limited-featured instance)
Data size	Configurable from 4 to 32 bits	8/16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length 9/17 bits
FIFO size	16 x 8 bits	8 x 8 bits
FIFO threshold	1 - 16 data	1 - 4 data
Number of data control	Up to 65535	Up to 1023, no remaining data counter (CTSIZE)
Autonomous in Stop mode with wake-up capability	Yes	Yes
Autonomous in Standby mode with wake-up capability	No	No

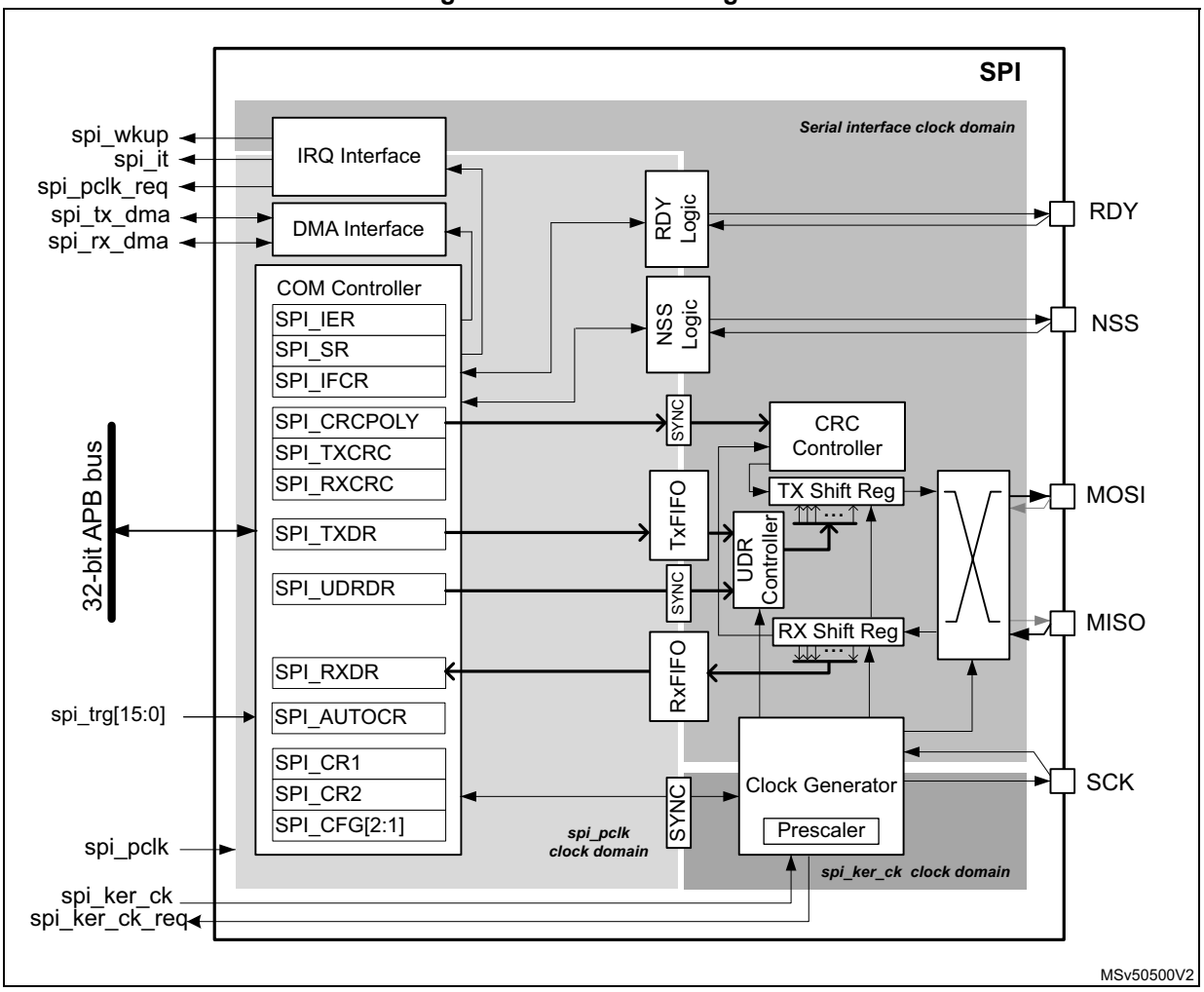
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 520: SPI wake-up and interrupt requests and Table 81: Functionalities depending on the working mode.

49.4 SPI functional description

49.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 641*.

Figure 641. SPI block diagram



The simplified scheme of *Figure 641* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 49.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 49.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

49.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 49.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 515* and *Table 516* for the list of SPI input / output pins and internal signals.

Table 515. SPI input/output pins

Pin name	I/O type	Description
MISO(1)	Input/output	Master data input / slave data output
MOSI(1)	Input/output	Master data output / slave data input
SCK	Input/output	Master clock output / slave clock input
NSS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output

1. Functionality of MOSI and MISO pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 516. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request
spi_trg[15:0]	Input	SPI triggers

Description of SPI interconnections

Table 517. SPI interconnection (SPI1 and SPI2)

Signal name	Trigger source
spi_trg0	gpdma1_ch0_tcf
spi_trg1	gpdma1_ch1_tcf
spi_trg2	gpdma1_ch2_tcf
spi_trg3	gpdma1_ch3_tcf
spi_trg4	exti4
spi_trg5	exti9
spi_trg6	lptim1_ch1
spi_trg7	lptim2_ch1
spi_trg8	comp1_out
spi_trg9	comp2_out
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12	-
spi_trg13	-
spi_trg14	-
spi_trg15	-

Table 518. SPI interconnection (SPI3)

Signal name	Trigger source
spi_trg0	gpdma1_ch0_tcf
spi_trg1	gpdma1_ch1_tcf
spi_trg2	gpdma1_ch2_tcf
spi_trg3	gpdma1_ch3_tcf
spi_trg4	exti4
spi_trg5	exti8
spi_trg6	lptim1_ch1
spi_trg7	lptim3_ch1
spi_trg8	comp1_out
spi_trg9	comp2_out
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12	-
spi_trg13	-
spi_trg14	-
spi_trg15	-

49.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

49.4.4 Communications between one master and one slave

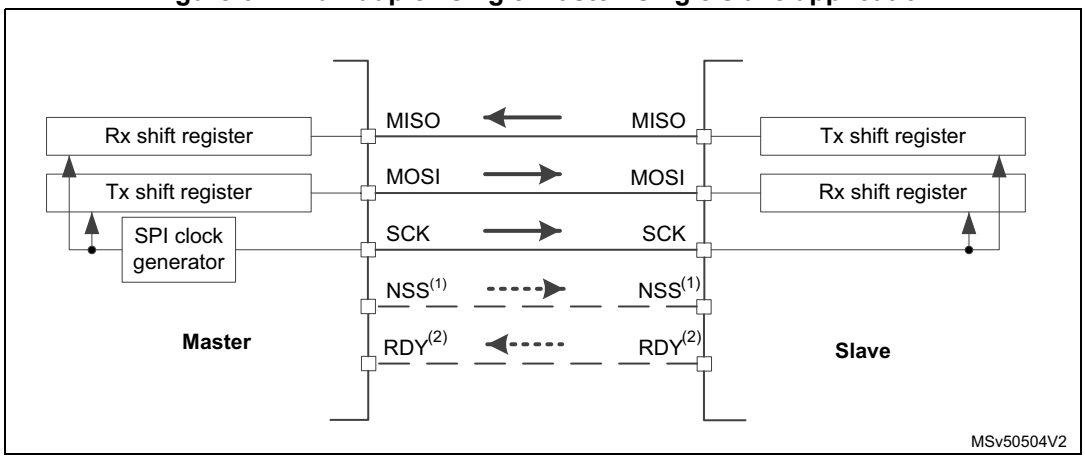
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 642. Full-duplex single master/ single slave application

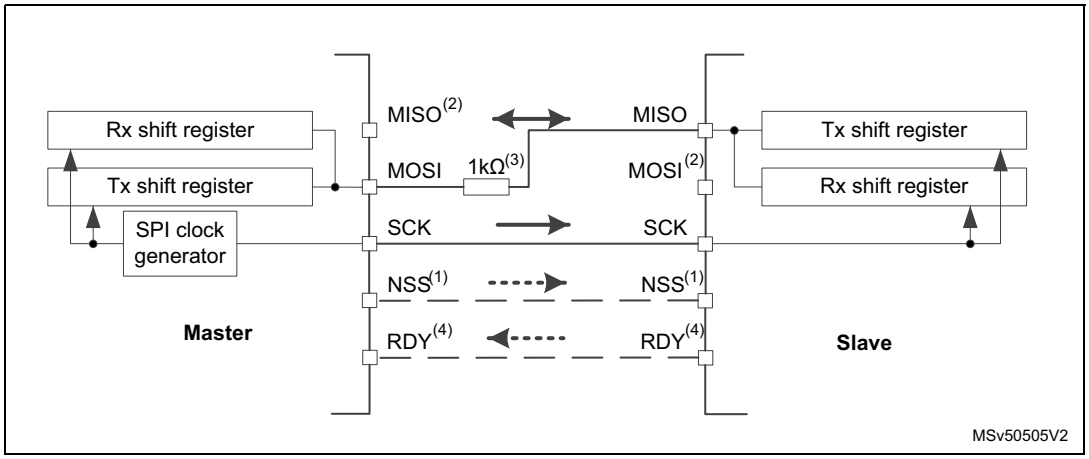


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 49.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 643. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 49.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 49.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

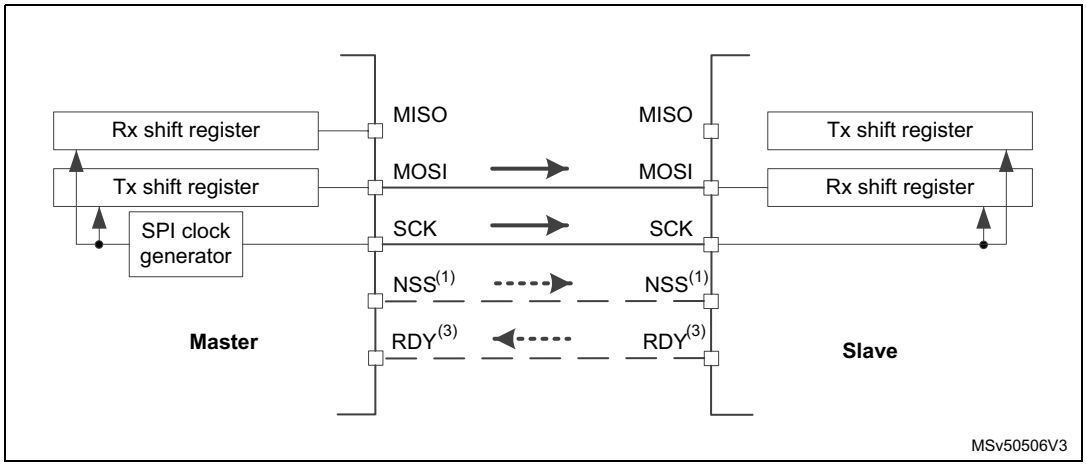
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 49.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 644. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 49.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

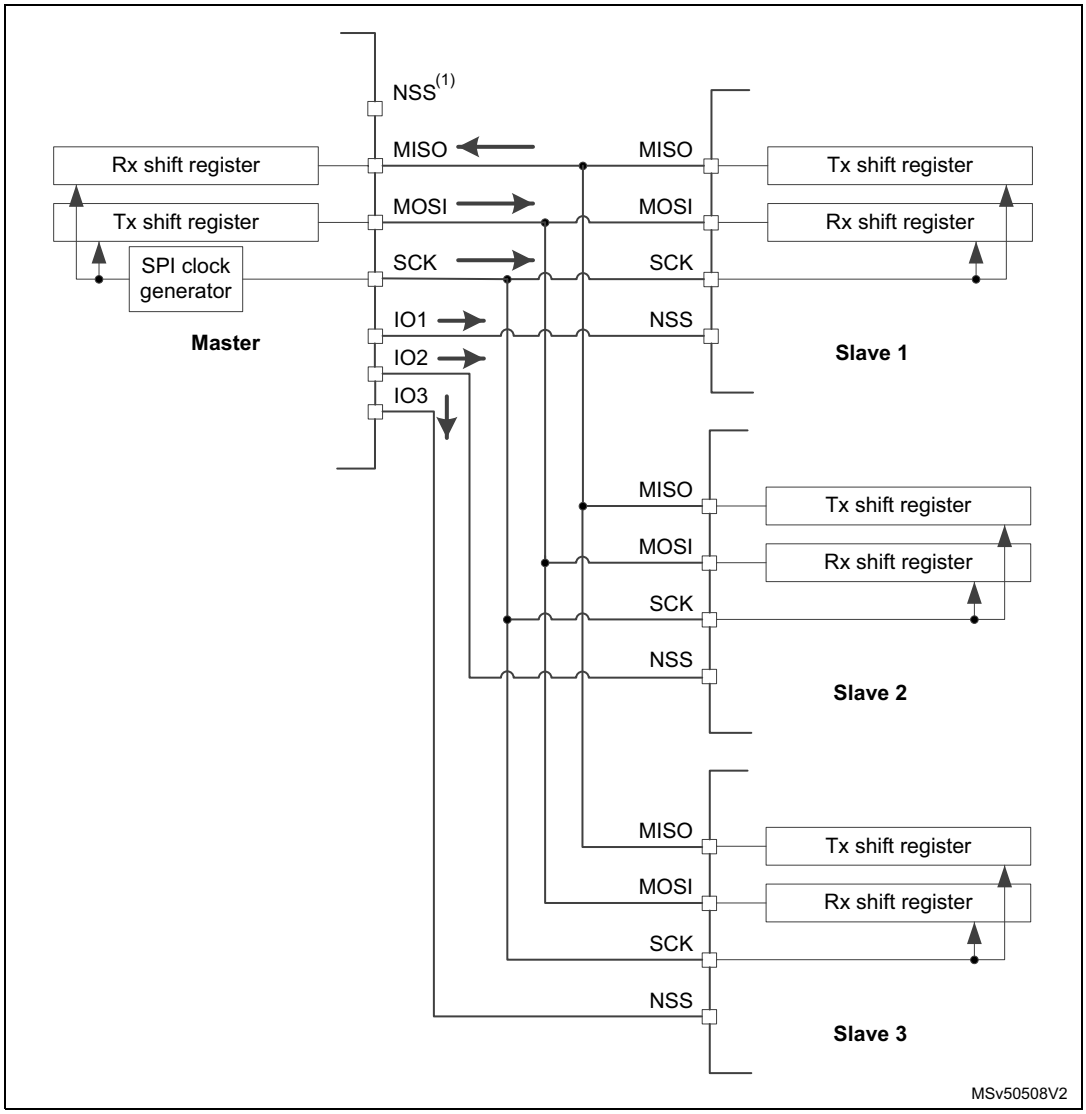
49.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 645.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 645. Master and three independent slaves connected in star topology



1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 49.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

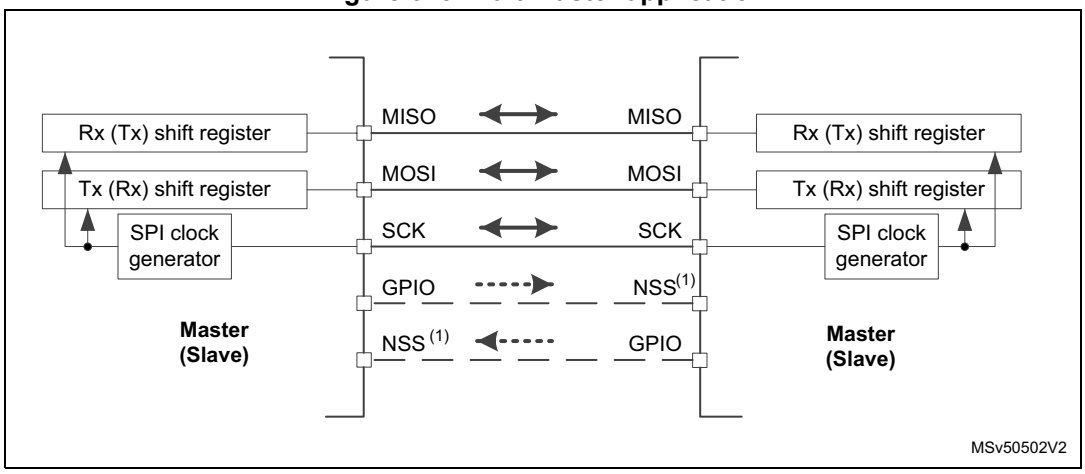
49.4.6 Multimaster communication

Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 646. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

49.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the

proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

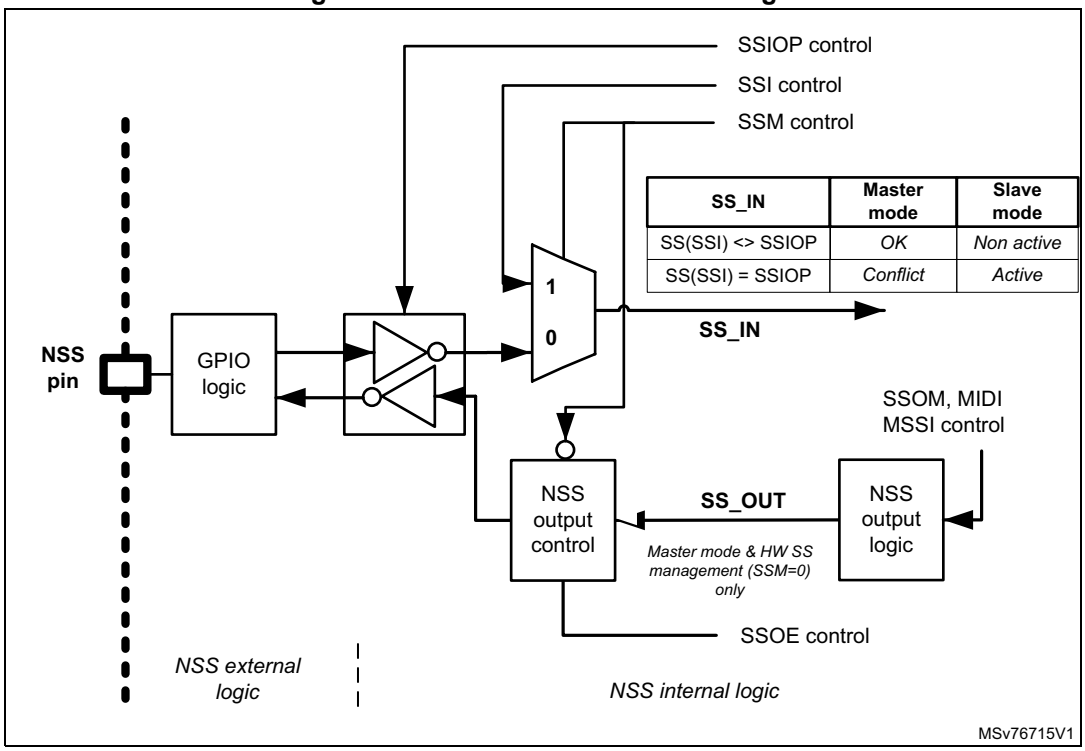
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

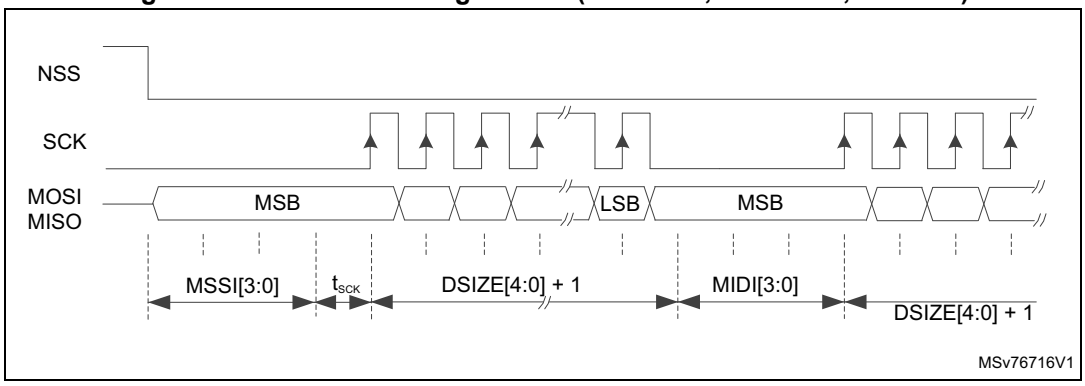
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 647. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 648*).

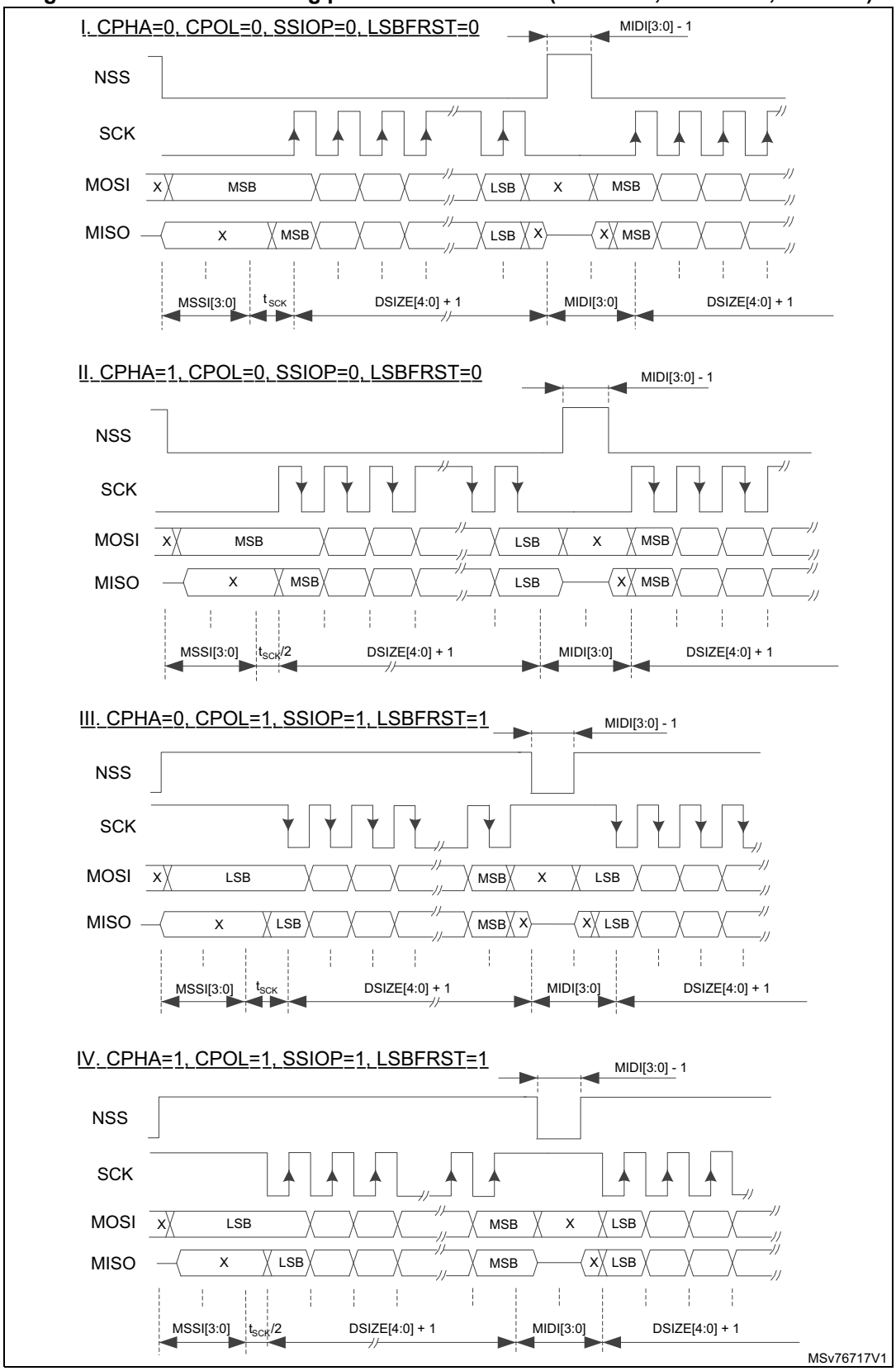
Figure 648. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 649*.

Figure 649. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

49.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

49.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

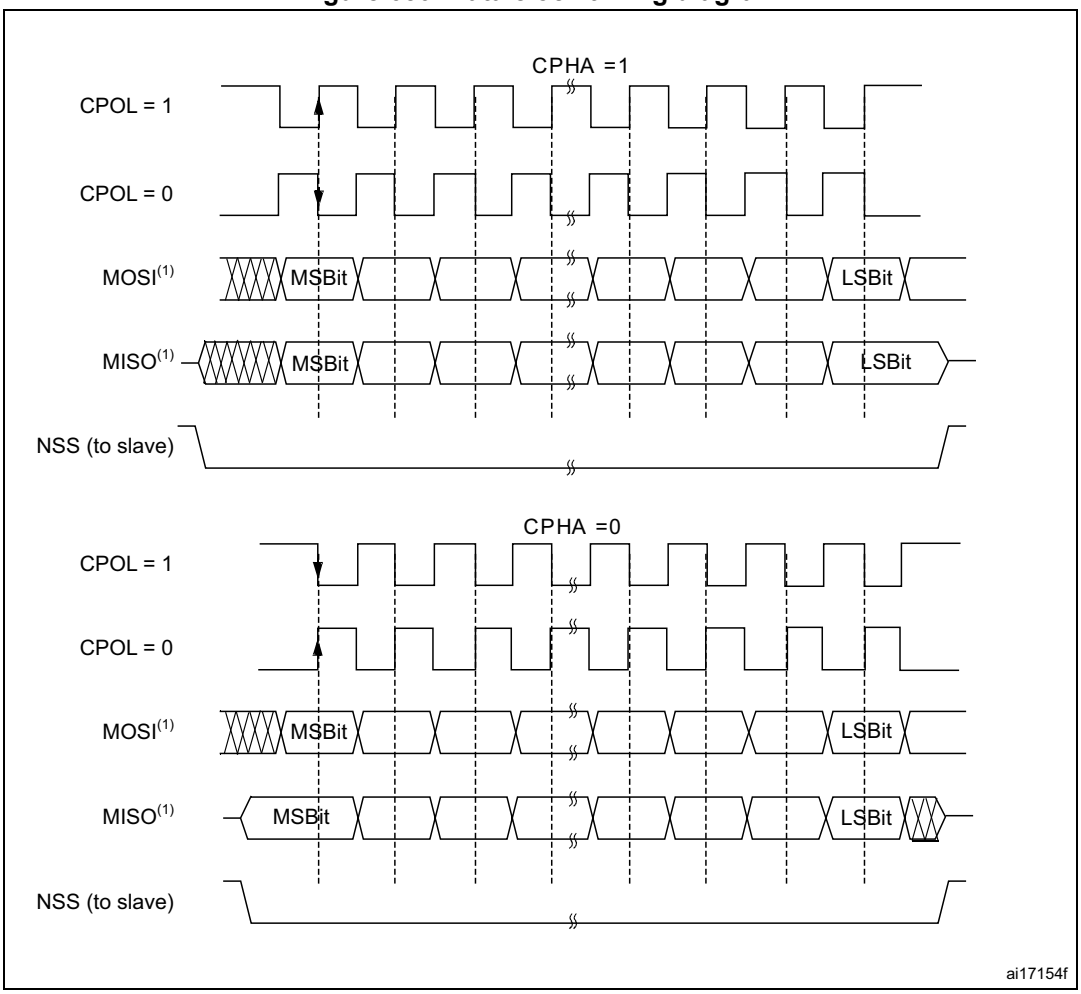
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 650*).

Figure 650 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 650. Data clock timing diagram



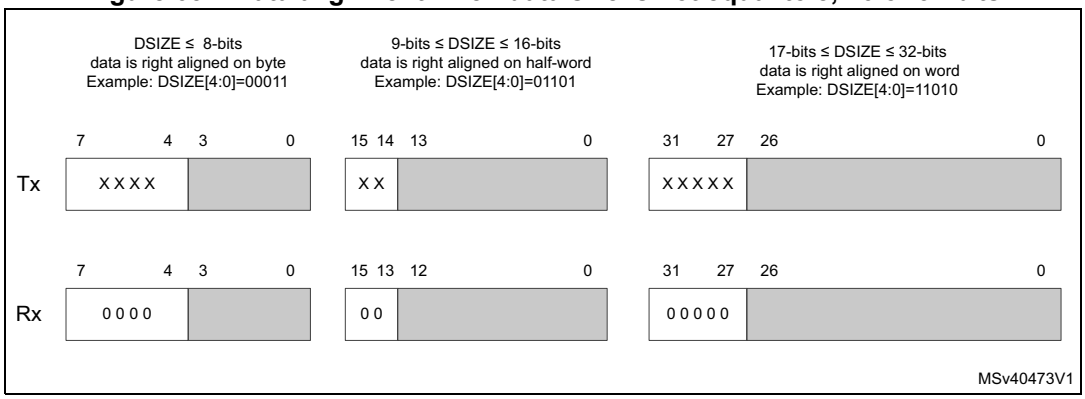
1. The order of data bits depends on the LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 49.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 651*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 49.4.12: SPI data transmission and reception procedures*.

Figure 651. Data alignment when data size is not equal to 8, 16 or 32 bits



49.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 49.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

49.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 49.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 49.4.14*.

49.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

49.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

49.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

49.4.15 Autonomous mode

The SPI is capable of handling and initializing transfers autonomously. It requires no specific system execution interaction until the ongoing transfer ends.

Such autonomous transfers can be handled not only in Run or Sleep mode but even in Stop mode when the SPI logic is able to provide temporary clock requests addressed to the reset

and clock controller (RCC) to ensure clocking of those SPI domains just necessary for handling the data flow between the memory and the peripheral interface at dependency in the SPI mode.

In Stop mode, the APB clock is requested by the peripheral each time the SPI registers need to be updated based on specific traffic events (mainly TXP and RXP). The required clock is provided by RCC if SPI autonomous mode is enabled at the RCC configuration and the SPI is clocked by an internal oscillator available in Stop mode.

Interrupts or DMA requests are then generated, depending on the SPI configuration. If no interrupt is enabled, the device remains in Stop mode. If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the device remains in Stop mode. If an enabled interrupt occurs, the device wakes up from Stop mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt is set. This is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low-power consumption especially, and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode.

Slave mode

When the SPI is configured as a standard slave and the device is at Stop mode, the SPI kernel clock and the SPI APB clock are not provided permanently. All the data flow between the SPI interface and associated FIFOs is handled by an external SCK clock provided by the outer master device within the serial interface clock domain. APB clock temporal requests are then based upon specific traffic events at dependency on the SPI configuration. As the slave never initializes a transfer, there is no need to synchronize any transfer start in this mode.

Note: The peripheral clock request stays pending until the flag with enabled interrupt stays set. This is why it is important to service these pending requests and clear their flag as soon as possible to achieve the lowest power consumption, and the application must acknowledge all pending interrupt events before switching the SPI to low-power mode.

Master mode

The SPI operating in master mode provides the SCK signal for outer slaves until the transfer is complete. The SCK signal is derived from the SPI clock generator running within the kernel clock domain fed from the RCC upon kernel clock request provided by the SPI when the device is in Stop mode. Temporal requests for the APB clock are then based upon specific traffic events, depending on the SPI configuration. The SPI master always initializes a transfer.

To minimize consumption in Stop mode, it is suggested to combine communication starts triggered by hardware (the TRIGEN bit is set in the SPI_AUTOCR register) and transfers of predefined data size (TSIZE > 0 in the SPI_CR2 register). This ensures that any APB clock request is suppressed between EOT handling and the next trigger event.

A transfer starts once the CSTART bit is set. In the case of master transmitter, the TxFIFO must be filled by data, too. The CSTART bit can be written either by software or by hardware when a synchronous trigger is detected at Run, Sleep, or Stop mode. The trigger source is selected by the TRIGSEL bits and enabled by the TRIGEN bit of the SPI_AUTOCR register. When the enabled trigger is detected, the transfer starts and continues by handling data until the EOT event or the transfer suspension. When the TRIGEN bit is changed, the user must prevent any trigger event occurrence. If the user cannot prevent that, the TRIGEN bit

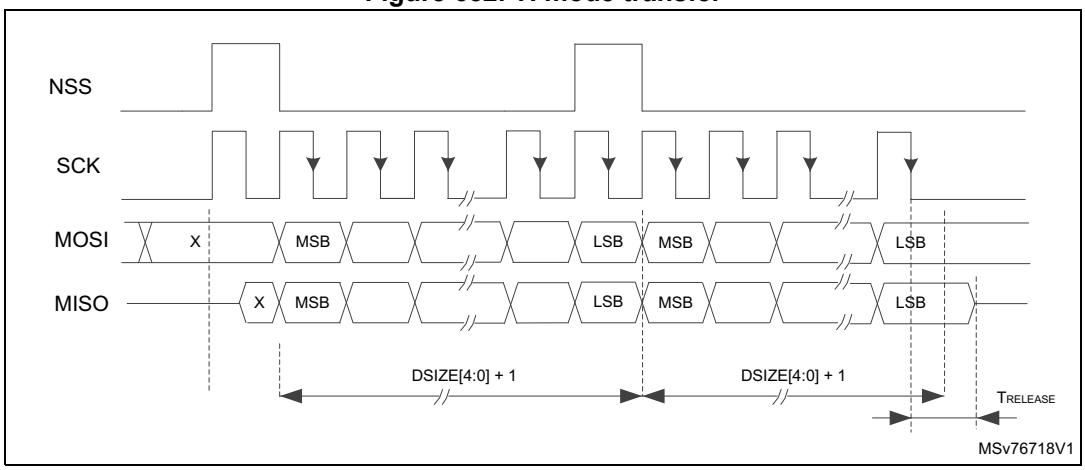
must be written while the SPI is disabled otherwise the peripheral behavior is not guaranteed.

49.5 SPI specific modes and control

49.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 652*.

Figure 652. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

49.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

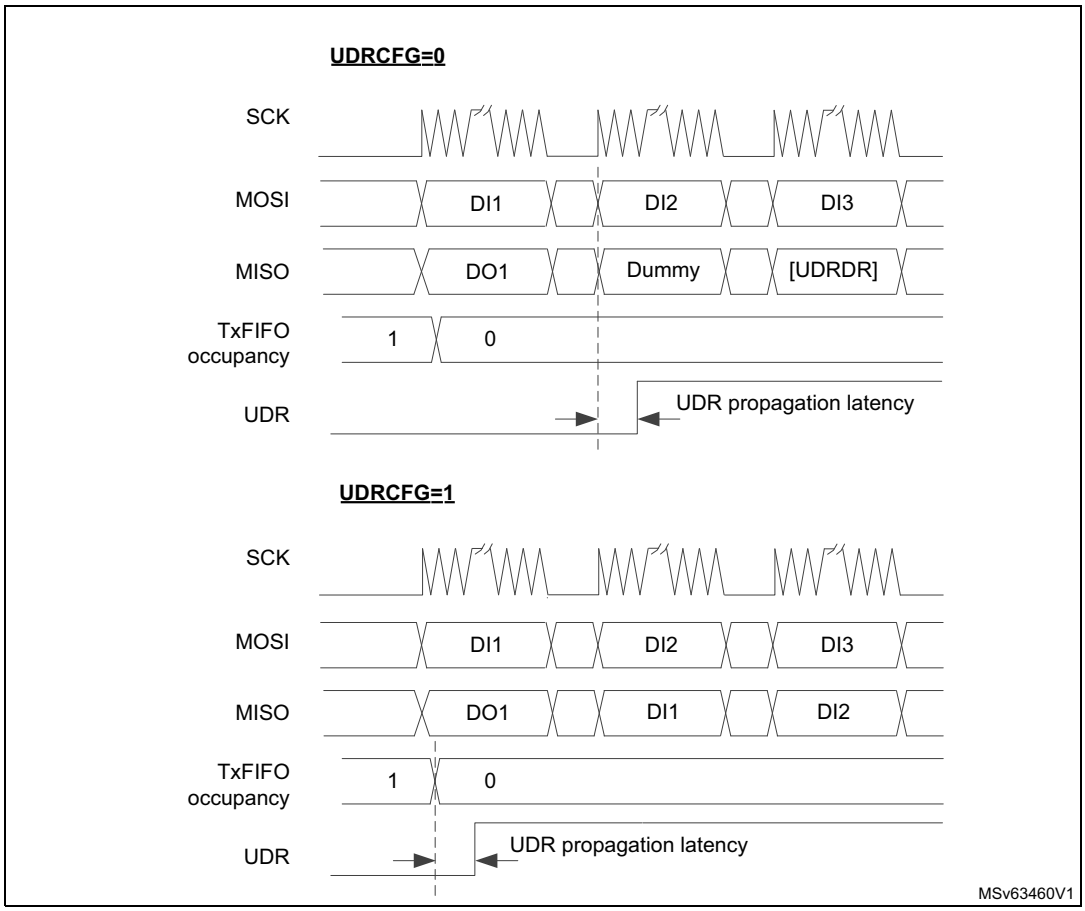
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 653*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the lastly received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 653. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

49.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 49.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

49.6 SPI in low-power modes

The SPI supports autonomous operation down to Stop mode, refer to *Section 49.4.15: Autonomous mode*.

Table 519. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The SPI registers content is kept. If the autonomous mode is enabled at RCC configuration, and SPI is clocked by an internal oscillator available in Stop mode, transfers are functional. The DMA requests are functional, and the interrupts in this mode cause the device to exit Stop mode.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 49.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

49.7 SPI interrupts

Table 520 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all

the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 520. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability(1)(2)
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC(3)		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 49.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

49.8 SPI registers

49.8.1 SPI control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be set by software only when SPI is disabled (SPE = 0). It is cleared by hardware exclusively whenever the SPE bit is changed from 1 to 0, either by hardware or software.

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI communication. it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, part of the SPI_AUTOCR register and IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

49.8.2 SPI control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

49.8.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator

0: bypass is disabled

1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 49.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.
The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.
00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits
Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.
The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 653: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats lastly received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8bit, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

49.8.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.
0000: no delay
0001: 1 clock cycle period delay
...
1111: 15 clock cycle periods delay
Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

49.8.5 SPI interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

49.8.6 SPI status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus or during autonomous operation in low-power mode.
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

49.8.7 SPI interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

49.8.8 SPI autonomous mode control register (SPI_AUTOCR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Hardware control of CSTART triggering enable

0: Hardware control disabled

1: Hardware control enabled

Note: if user cannot prevent trigger event during write, the TRIGEN must be changed when SPI is disabled

Bit 20 **TRIGPOL:** Trigger polarity

0: trigger is active on raising edge

1: trigger is active on falling edge

Note: This bit can be written only when SPE = 0.

Bits 19:16 **TRIGSEL[3:0]**: Trigger selection (refer *Section : Description of SPI interconnections*).

0000: spi_trg0 is selected

0001: spi_trg1 is selected

...

1111: spi_trg15 is selected

Note: these bits can be written only when SPE = 0.

Bits 15:0 Reserved, must be kept at reset value.

49.8.9 SPI transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

49.8.10 SPI receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: data is always right-aligned. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

49.8.11 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.8.12 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

49.8.13 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

49.8.14 SPI underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

49.8.15 SPI register map

Table 521. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]				
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	SPI_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0																
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 49.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50 Serial audio interface (SAI)

50.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

50.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception
- Interrupt sources when enabled:
  - Errors

  - FIFO requests
- 2-channel DMA interface

50.3 SAI implementation

Table 522. SAI features(1)

SAI features	SAI1
I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC'97	X
FIFO size	8 words
SPDIF	X
PDM	X(2)

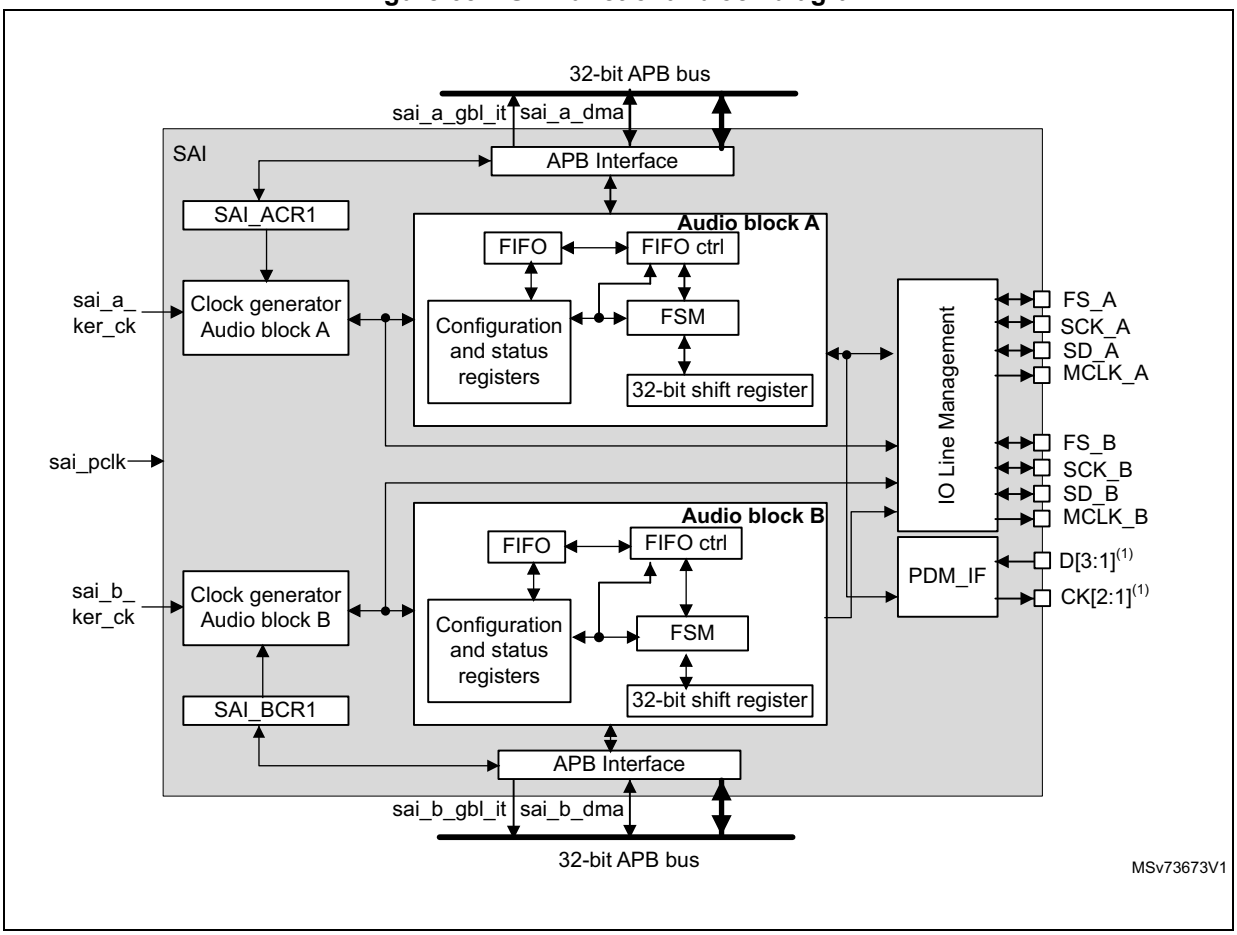
1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

50.4 SAI functional description

50.4.1 SAI block diagram

Figure 654 shows the SAI block diagram while *Table 523* and *Table 524* list SAI internal and external signals.

Figure 654. SAI functional block diagram



The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

50.4.2 SAI pins and internal signals

Table 523. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 524. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock
SAI_MCLK_A/B	Output	Audio block A/B master clock
SAI_SD_A/B	Input/output	Data line for block A/B
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B
SAI_CK[4:1]	Output	PDM bitstream clock
SAI_D[4:1]	Input	PDM bitstream data

50.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC'97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

50.4.4 SAI synchronization mode

SAI subclock A and B can be synchronized.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one

for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

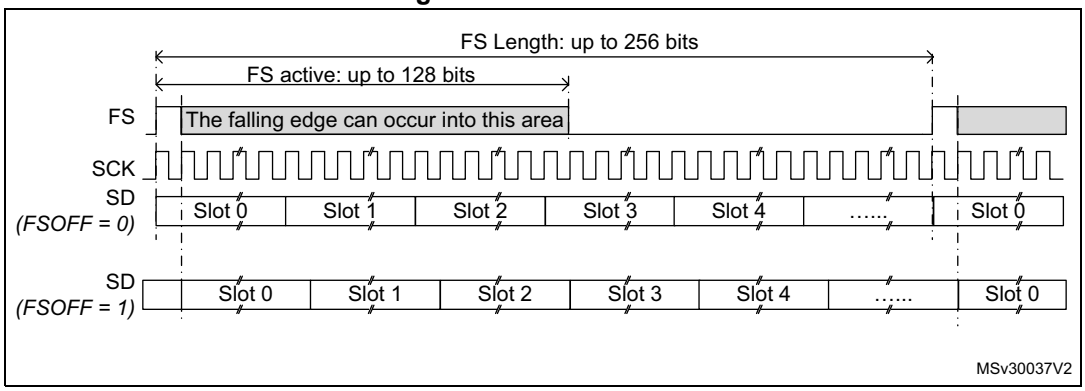
50.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

50.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 655* illustrates this flexibility.

Figure 655. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z

depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 50.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 50.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 50.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 50.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

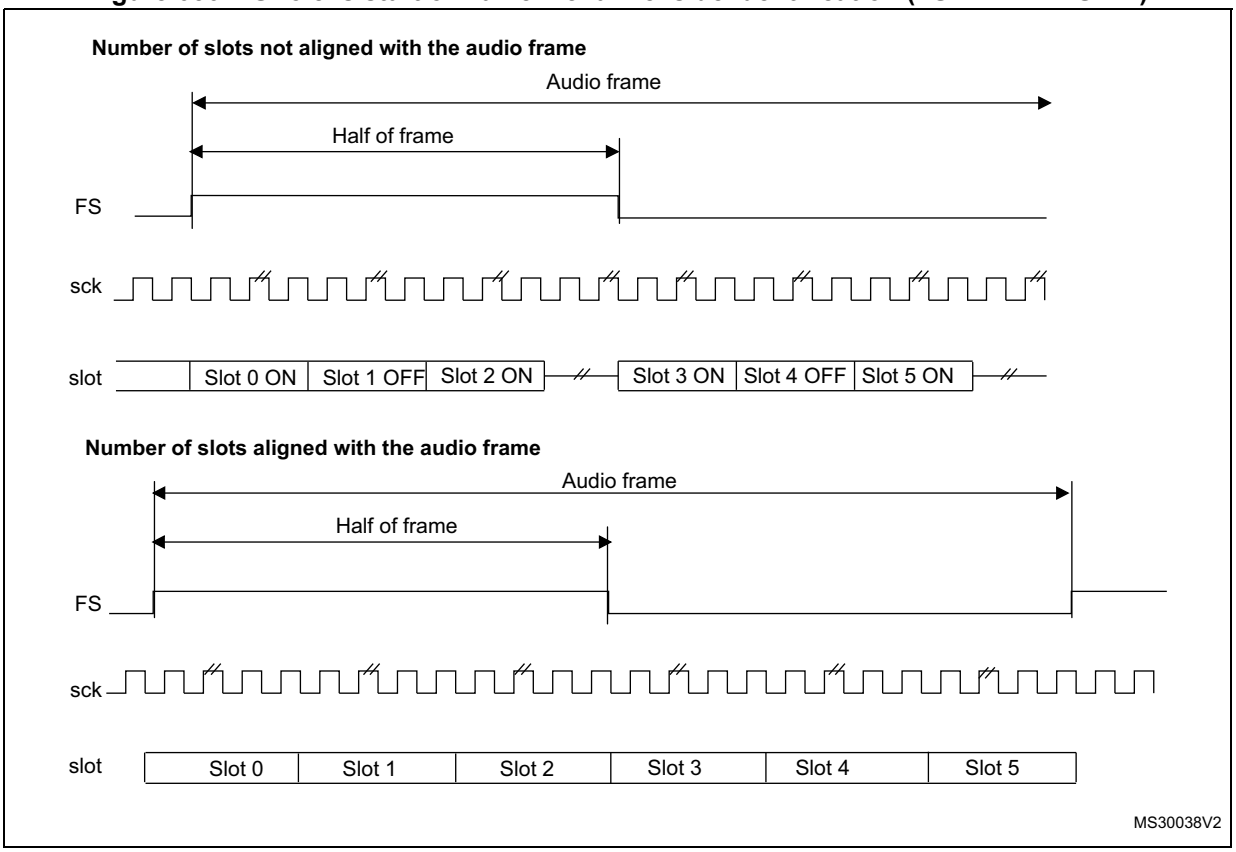
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 656. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

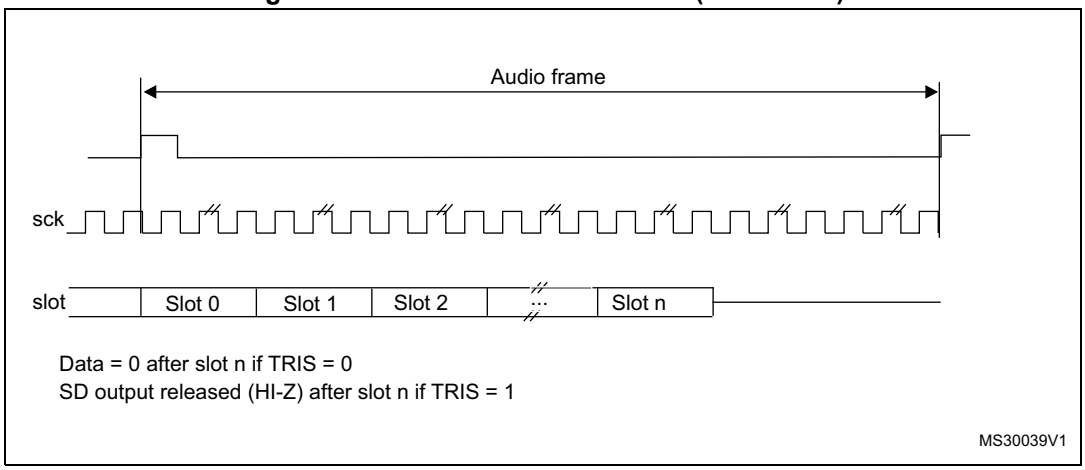


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 657. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

50.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

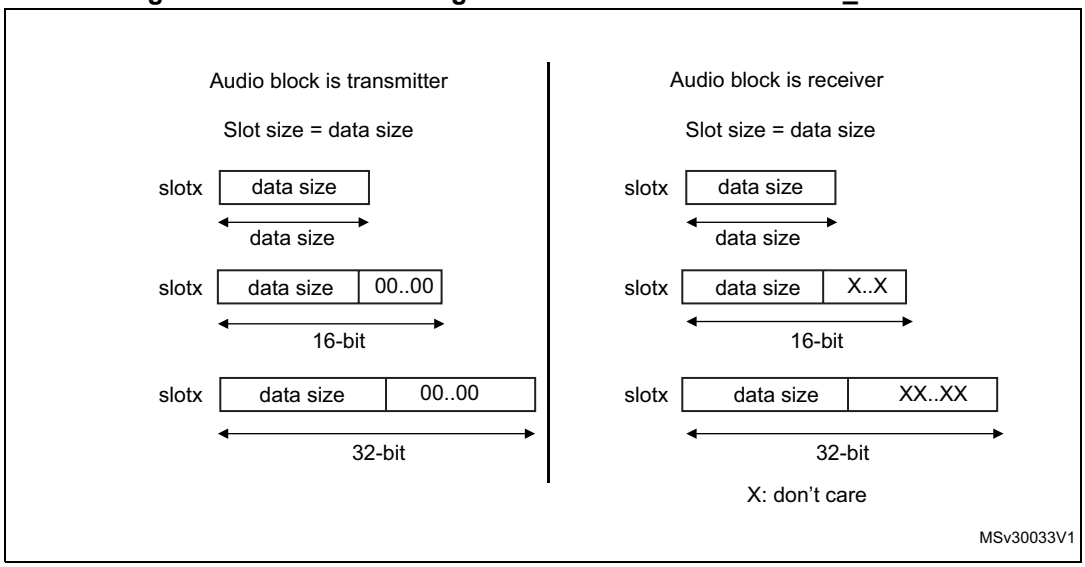
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

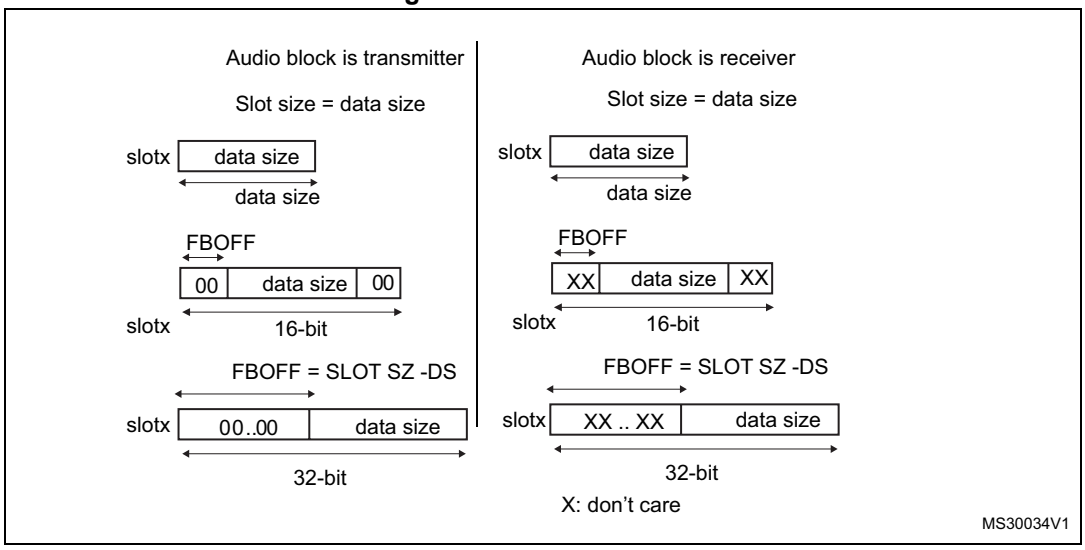
The slot size is also configurable as shown in *Figure 658.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 658. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 659. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤(SLOTSZ - DS)
- DS ≤SLOTSZ
- NBSLOT x SLOTSZ ≤FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 50.4.11: AC'97 link controller*.

50.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 525 shows MCLK_x activation conditions.

Table 525. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

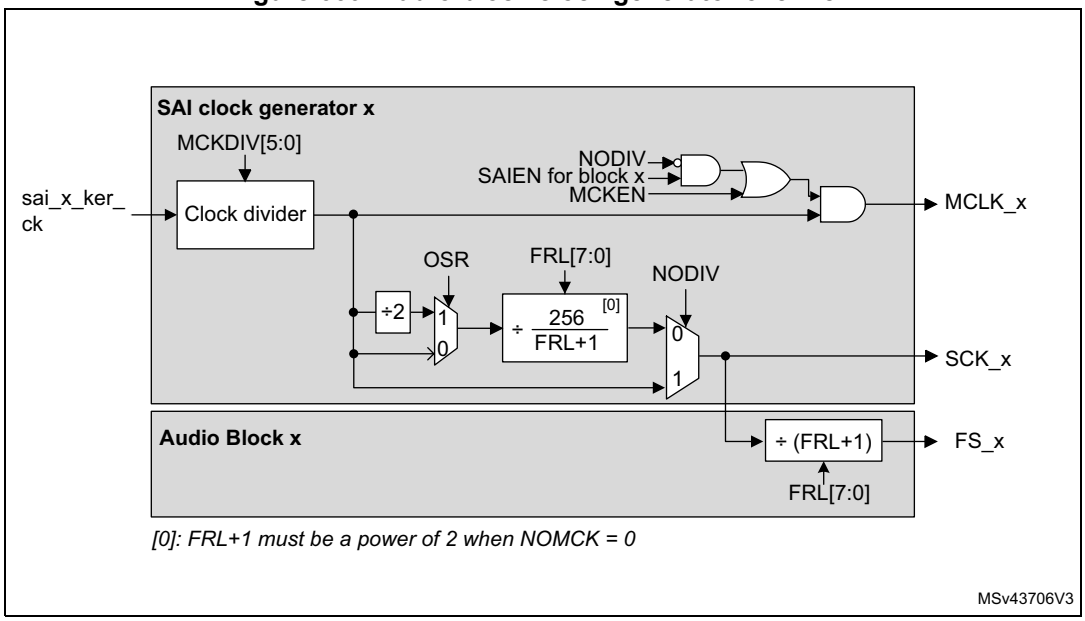
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 660 illustrates the architecture of the audio block clock generator.

Figure 660. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
- $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 526 gives programming examples for 48, 96, and 192 kHz.

Table 526. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio Sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

50.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

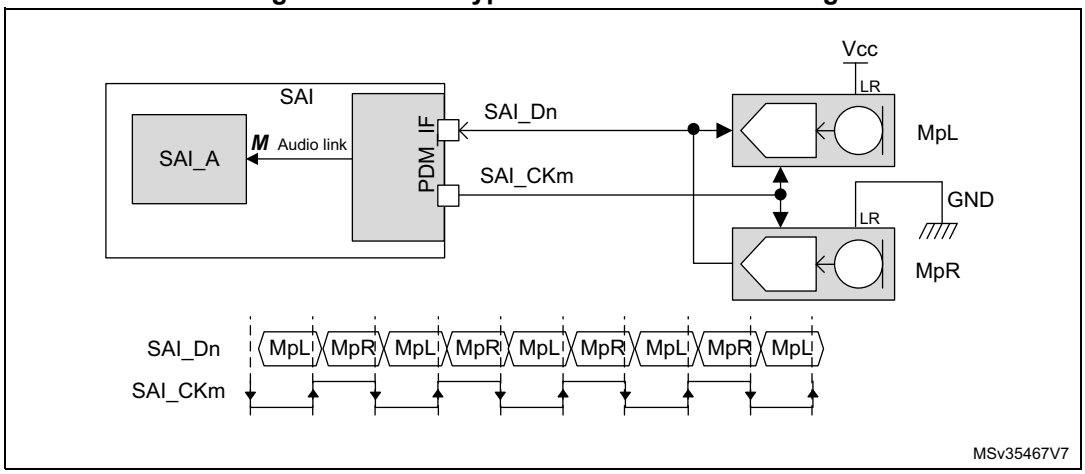
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

50.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 50.3: SAI implementation*).

Figure 661 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 661. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

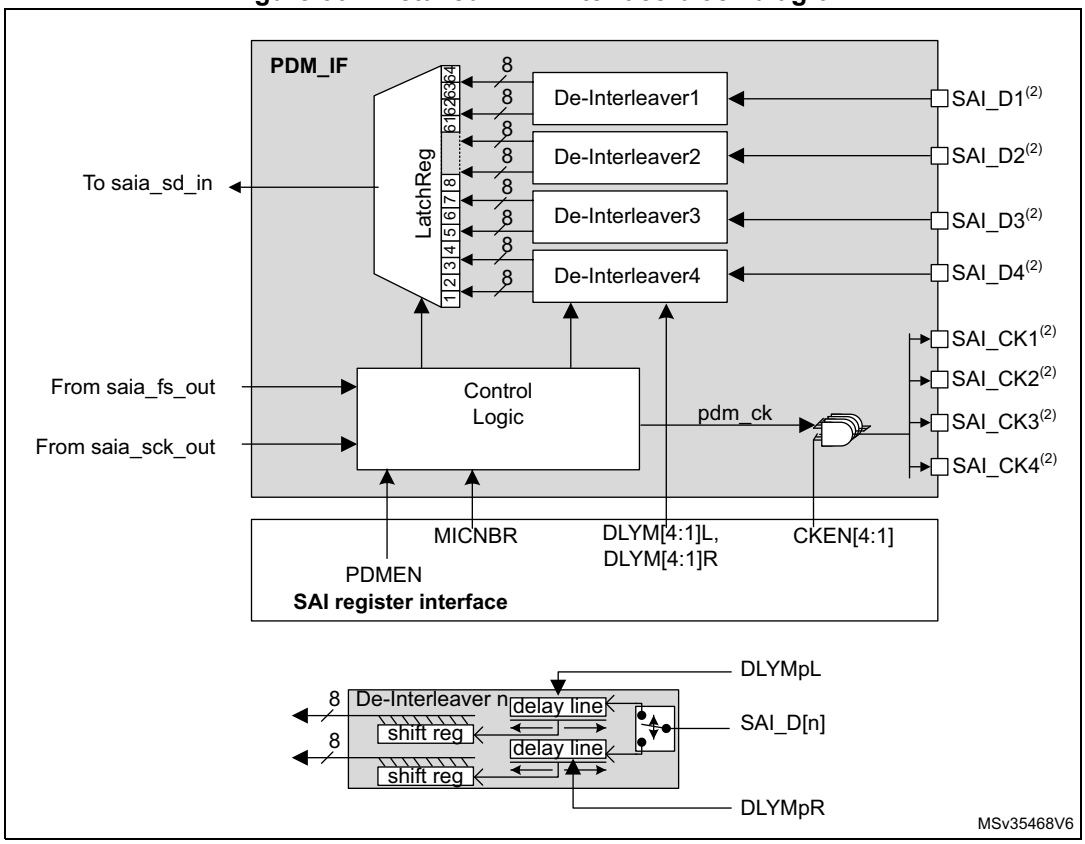
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 662 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 662. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 50.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose

between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

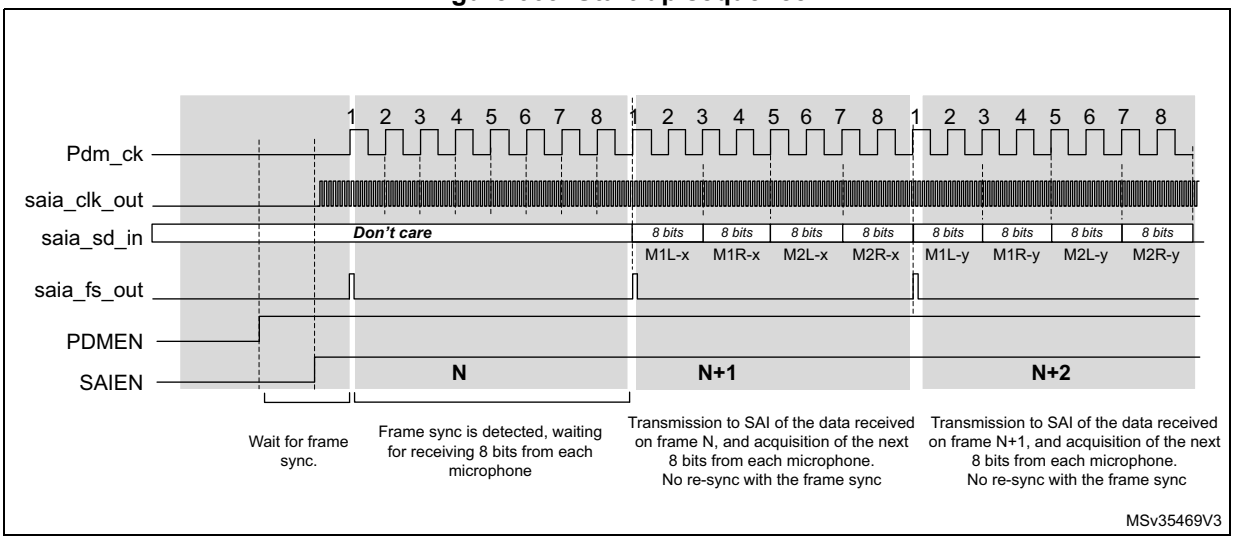
1. Configure SAI_A in TDM master mode (see *Table 527*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 663 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 663. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

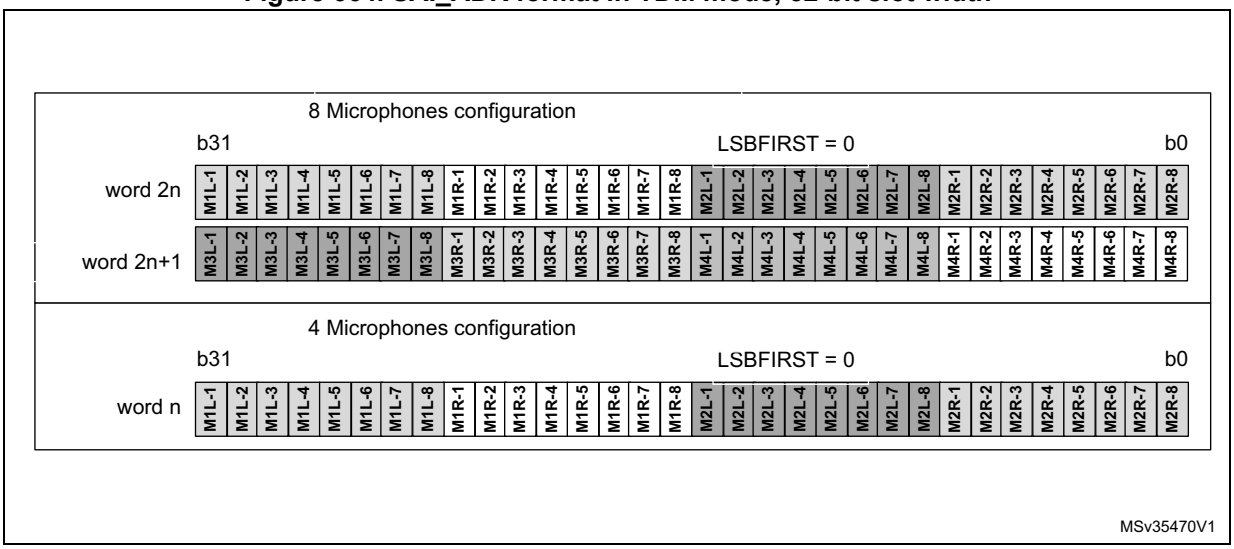
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 664*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 664. SAI_ADR format in TDM mode, 32-bit slot width

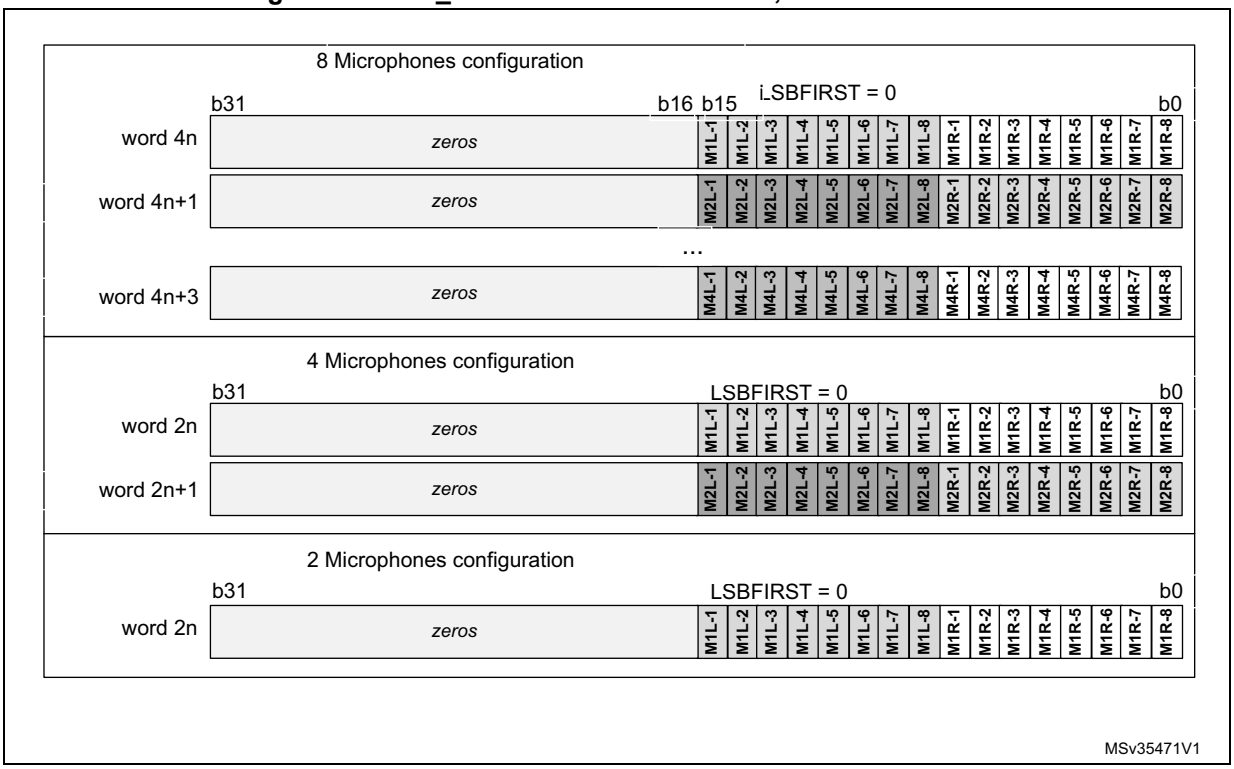


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 665*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 665. SAI_ADR format in TDM mode, 16-bit slot width

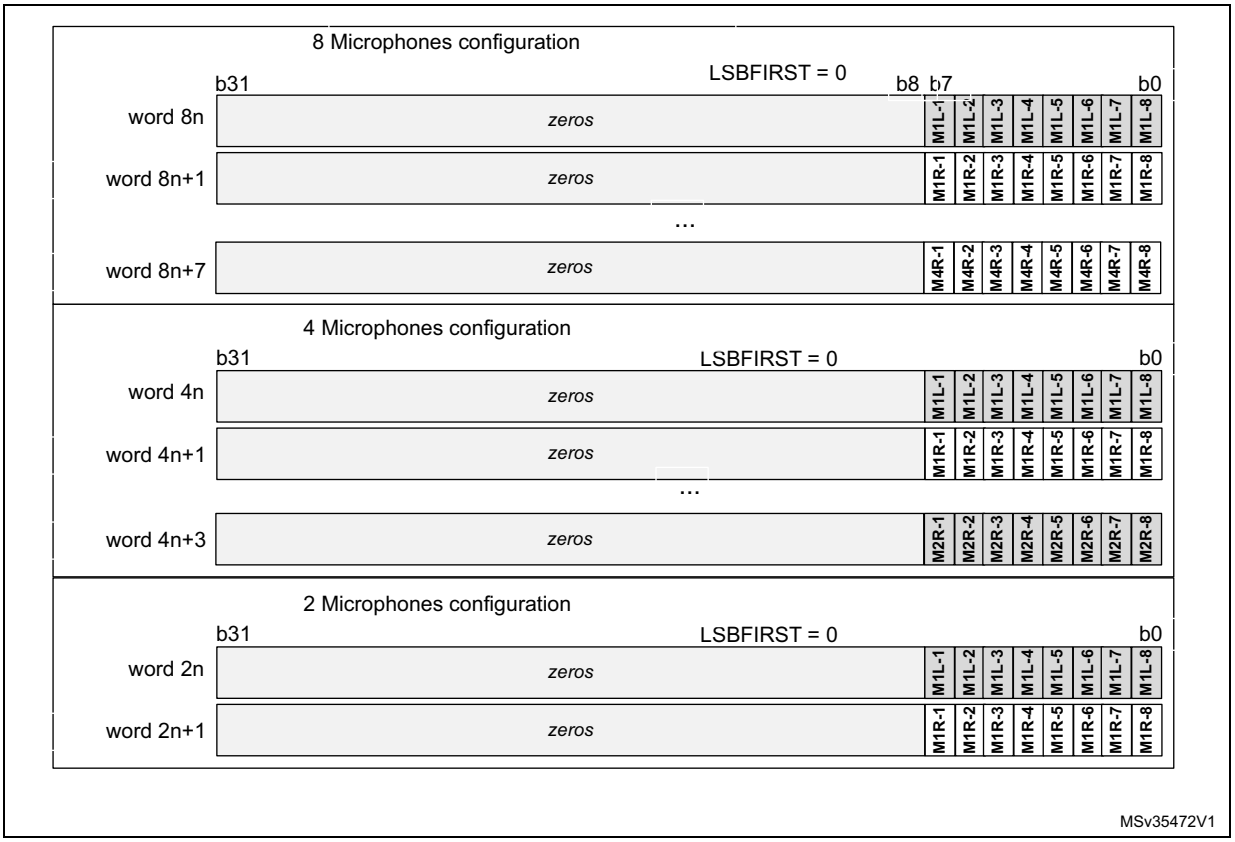


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 666*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 666. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 527* to ensure a good connection with the PDM interface.

Table 527. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 528*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 528*.
FSALL	0	Pulse width is one bit clock cycle.

Table 527. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 528*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 528*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 527*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 50.6.17*)

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 528. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 527: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device.*
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

50.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

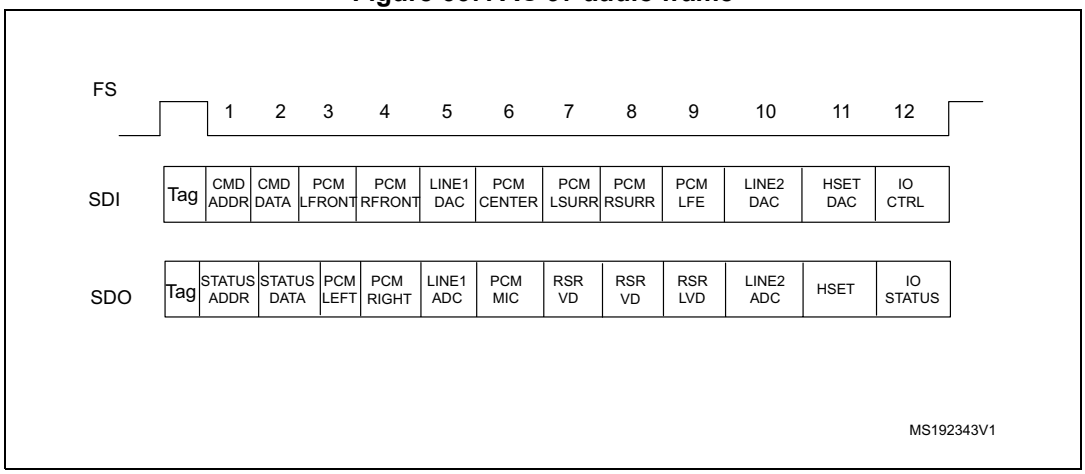
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 667 shows an AC’97 audio frame structure.

Figure 667. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 50.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

50.4.12 SPDIF output

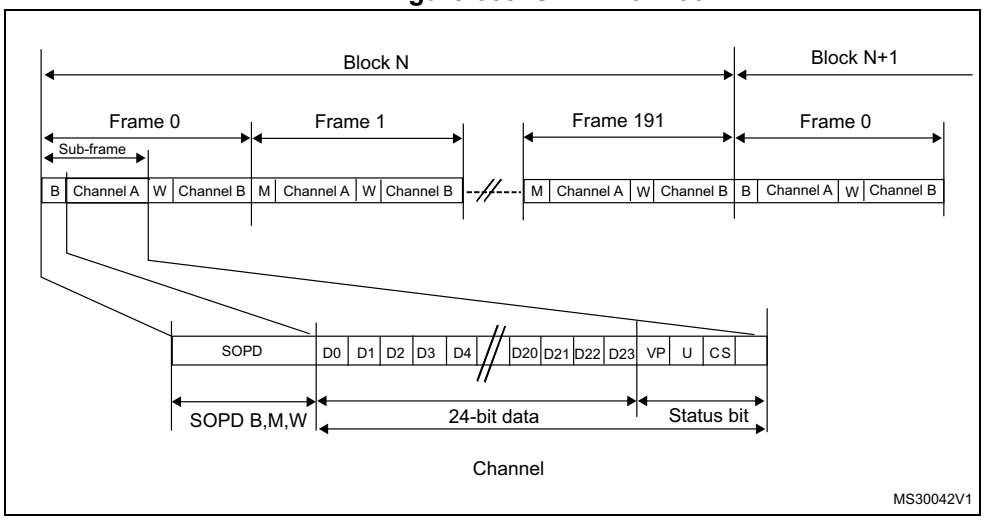
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 668*.

Figure 668. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 529*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 529. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

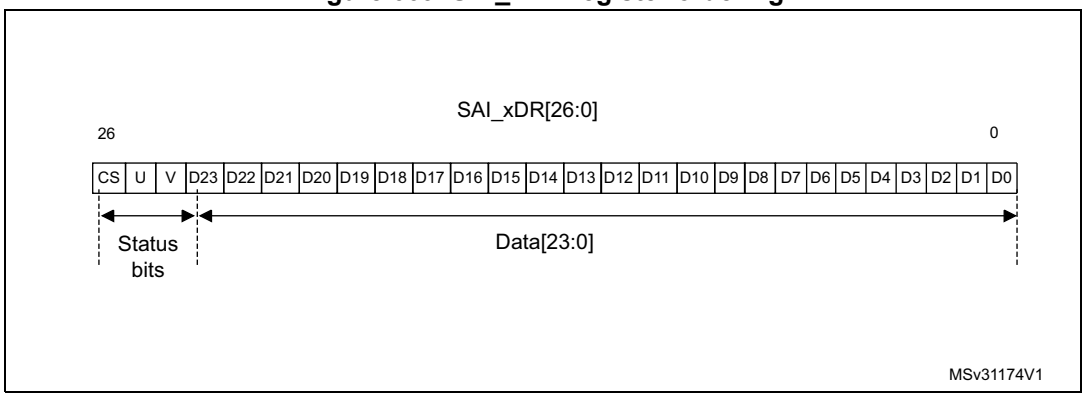
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 669. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 530*.

Table 530. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be

executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 531. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

50.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 670*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

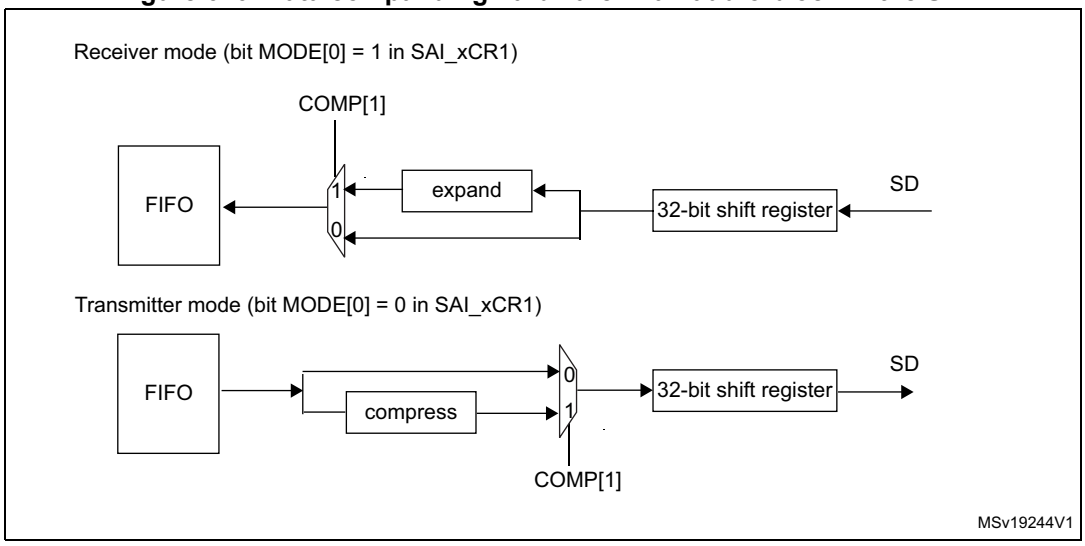
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1's complement or 2's complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 670. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

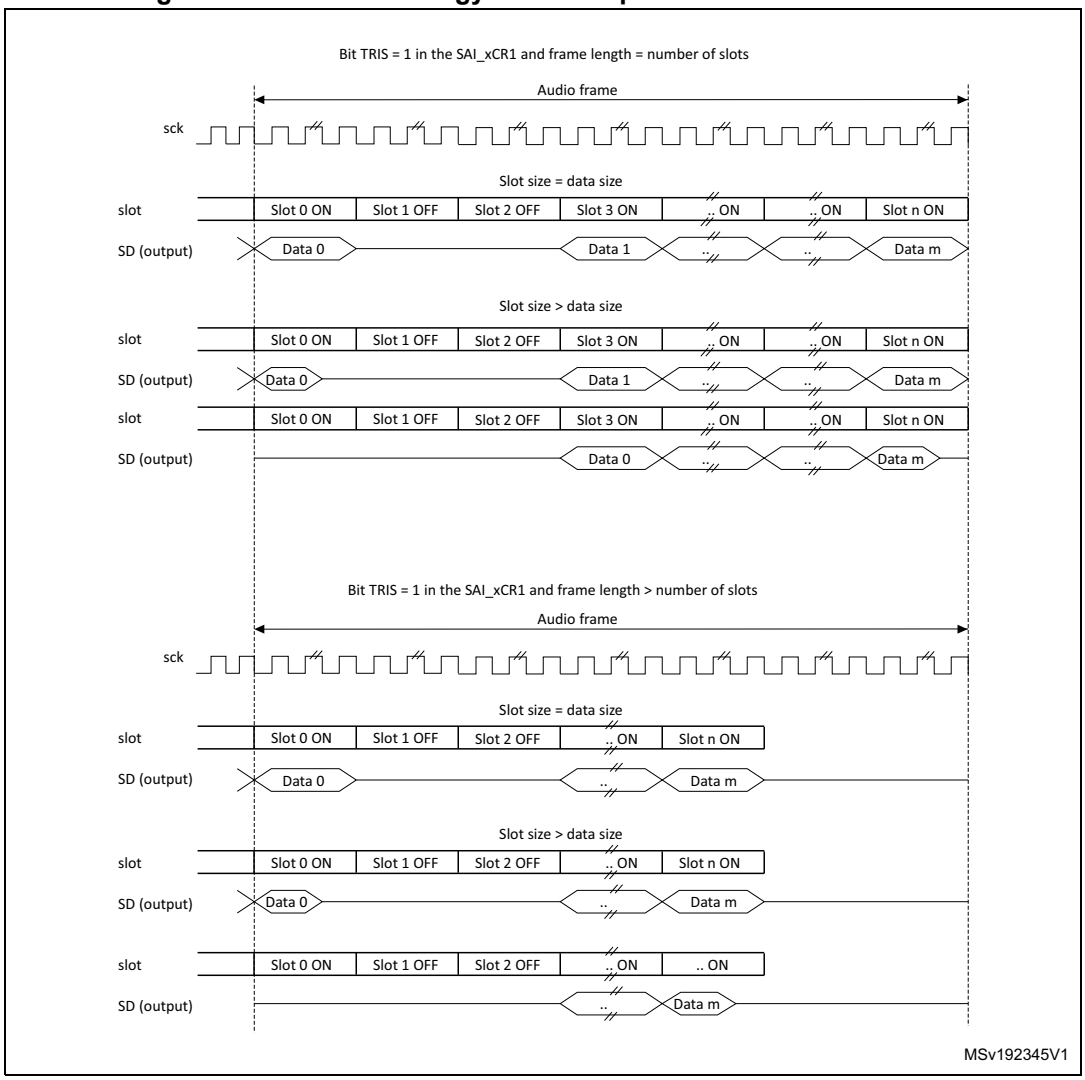
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

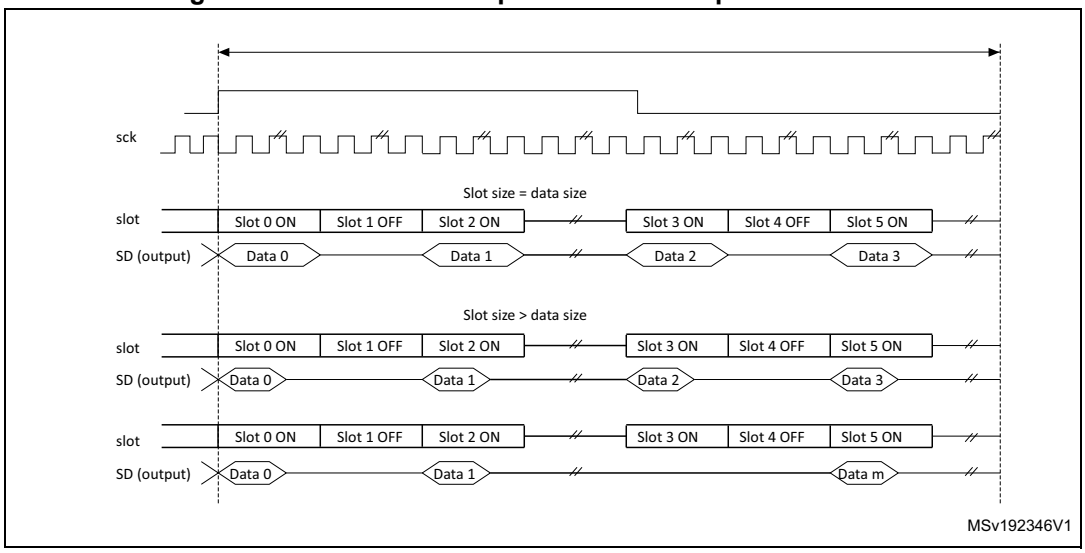
Figure 671 illustrates these behaviors.

Figure 671. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 672* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 672. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 671* and *Figure 672* are replaced by a drive with a value of 0.

50.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

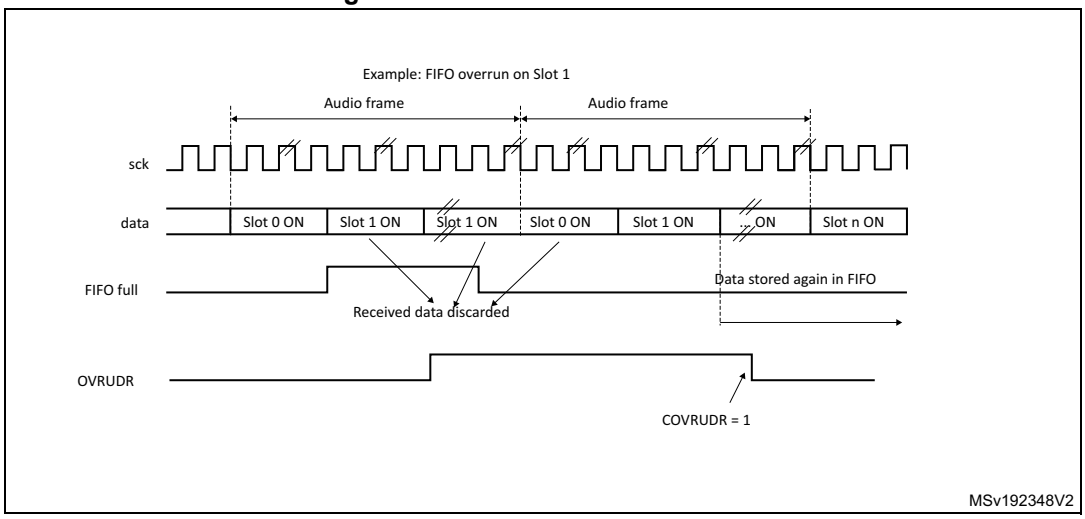
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 673*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 673. Overrun detection error



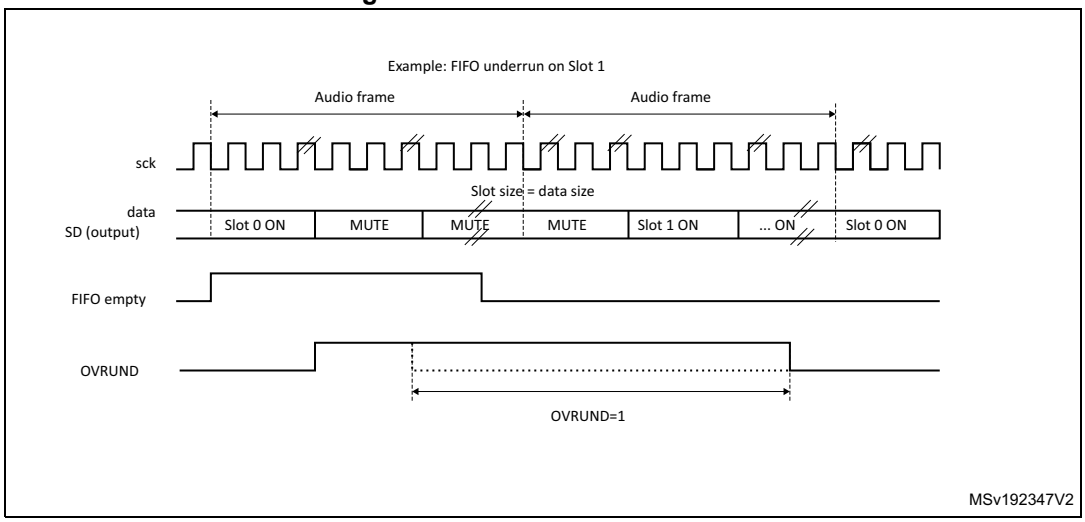
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 674*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 674. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

50.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

50.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 50.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

50.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 532*.

Table 532. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 50.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

50.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

50.6.1 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 50.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

50.6.2 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

50.6.3 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.

1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

50.6.4 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

50.6.5 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

50.6.6 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 50.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

50.6.7 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

50.6.8 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

50.6.9 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 50.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: Reserved.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

50.6.10 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

50.6.11 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

50.6.12 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

50.6.13 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

50.6.14 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 50.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

50.6.15 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

50.6.16 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

50.6.17 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
				rw	rw	rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **CKEN4**: Clock enable of bitstream clock number 4
This bit is set and cleared by software.
0: SAI_CK4 clock disabled
1: SAI_CK4 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK4 might not be available for all SAI instances. Refer to Section 50.3: SAI implementation for details.

Bit 10 **CKEN3**: Clock enable of bitstream clock number 3
This bit is set and cleared by software.
0: SAI_CK3 clock disabled
1: SAI_CK3 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK3 might not be available for all SAI instances. Refer to Section 50.3: SAI implementation for details.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2
This bit is set and cleared by software.
0: SAI_CK2 clock disabled
1: SAI_CK2 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK2 might not be available for all SAI instances. Refer to Section 50.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1
This bit is set and cleared by software.
0: SAI_CK1 clock disabled
1: SAI_CK1 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK1 might not be available for all SAI instances. Refer to Section 50.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones
This bit is set and cleared by software.
00: Configuration with 2 microphones
01: Configuration with 4 microphones
10: Configuration with 6 microphones
11: Configuration with 8 microphones
*Note: It is not recommended to configure this field when PDMEN = 1.**
The complete set of data lines might not be available for all SAI instances. Refer to Section 50.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable
This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.
Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.
0: PDM interface disabled
1: PDM interface enabled

50.6.18 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.
Note: This bitfield can be used only if D4 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 50.3: SAI implementation to check if it is available.

50.6.19 SAI register map

Table 533. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	Reserved																																
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0

Table 533. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																					0	0	0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

51 FD controller area network (FDCAN)

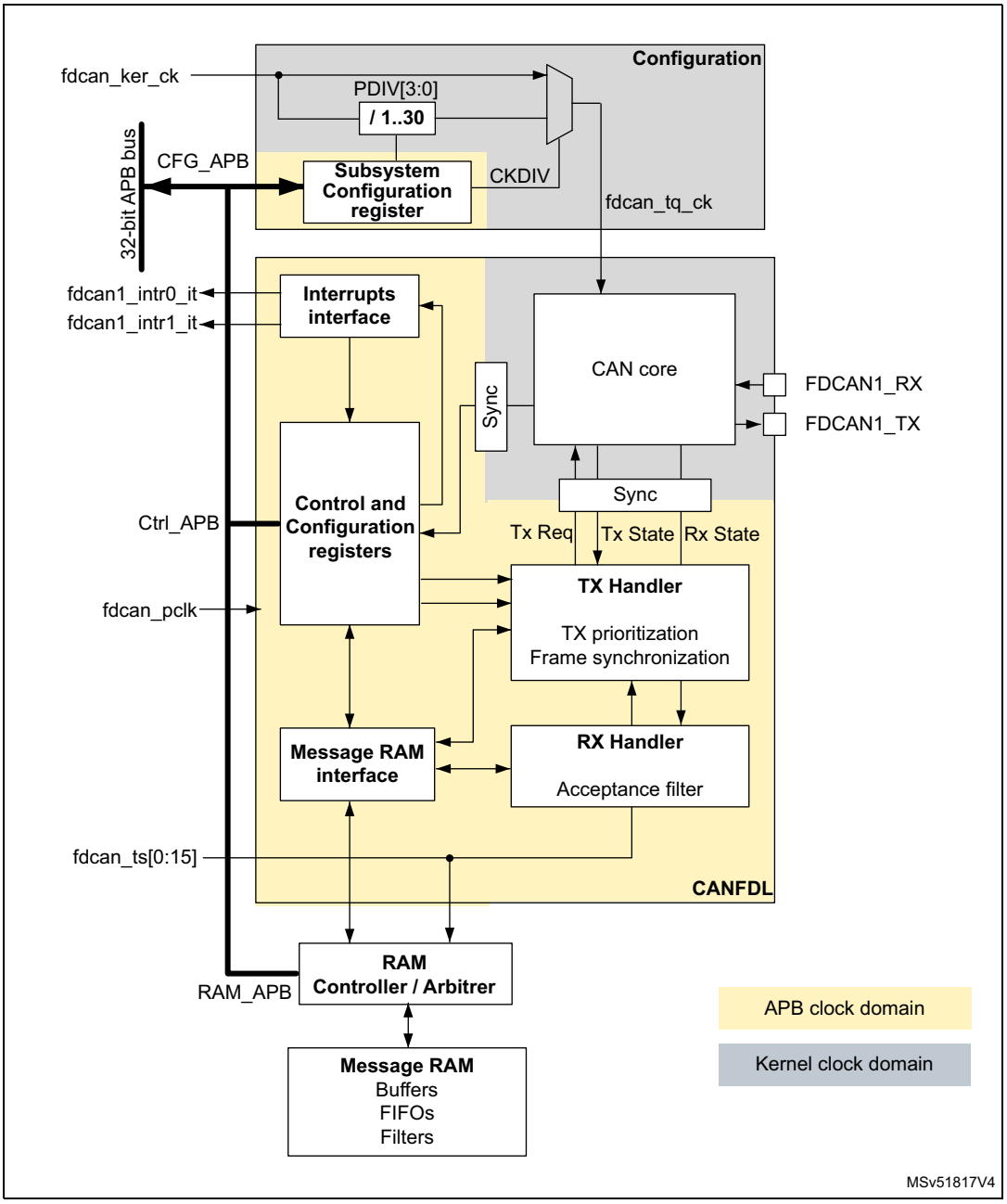
51.1 Introduction

The controller area network (CAN) subsystem (see *Figure 675*) consists of one CAN module, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 675. CAN subsystem.



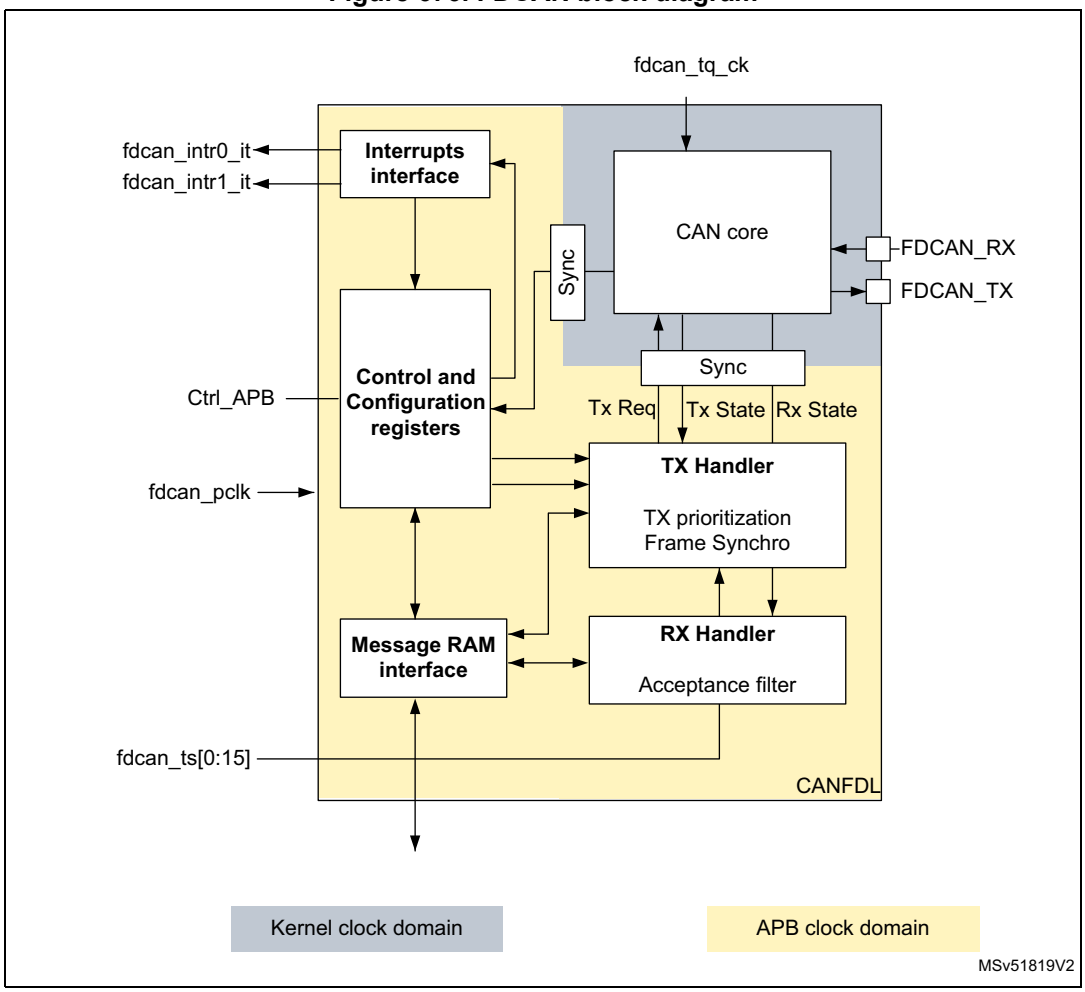
51.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

51.3 FDCAN functional description

51.3.1 FDCAN block diagram

Figure 676. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

51.3.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 534*, *Table 535*, and *Figure 675*.

Table 534. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_intr0_it	Digital output	FDCAN interrupt0
fdcan_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 535. CAN subsystem I/O pins

Name	Type	Description
FDCAN_RX	Digital input	FDCAN receive pin
FDCAN_TX	Digital output	FDCAN transmit pin

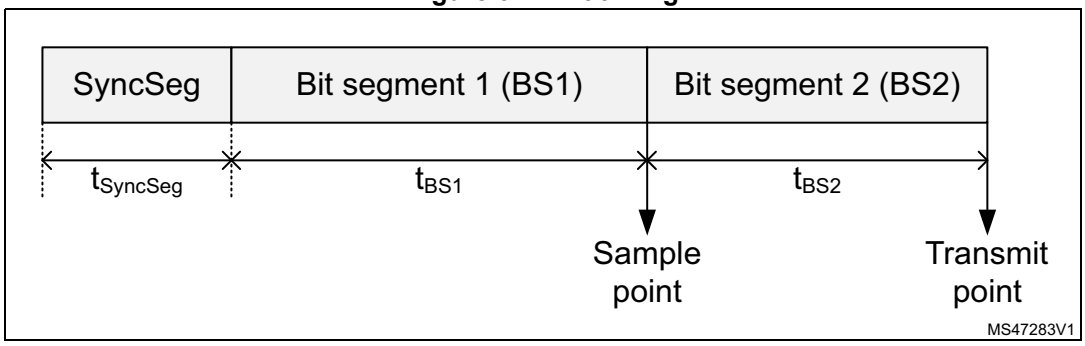
51.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 677*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 16 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 677. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 677*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time

$t_q = (DBRP[4:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$

$t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$

Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan_tq_ck} < f_{dcan_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

51.3.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 536*.

Table 536. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following

FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is

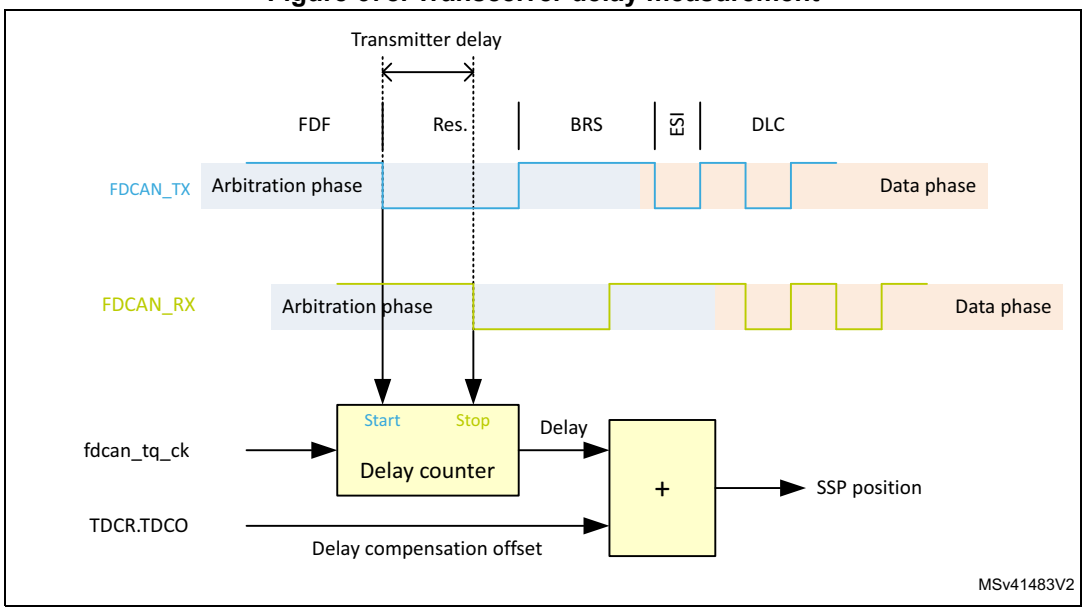
updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 678. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

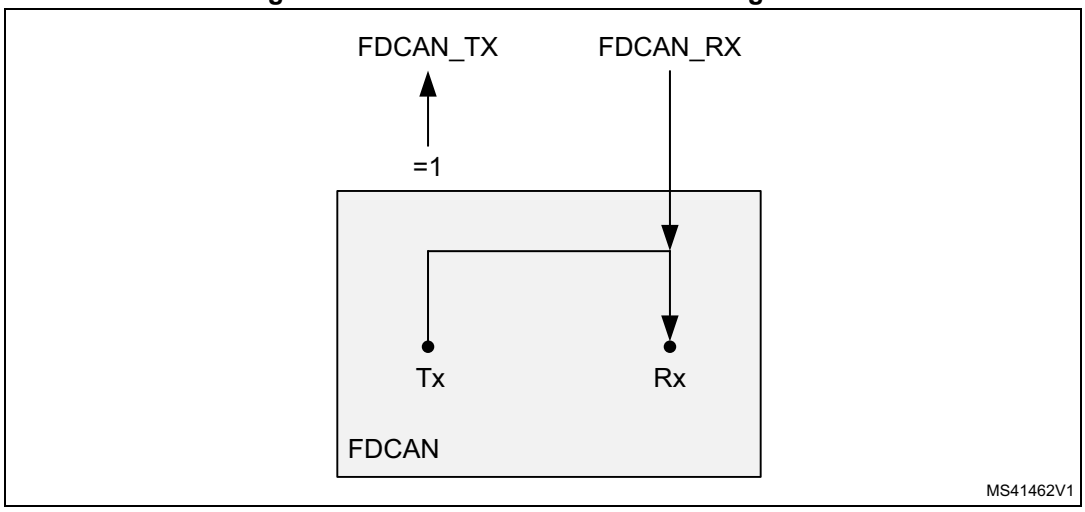
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 679* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 679. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

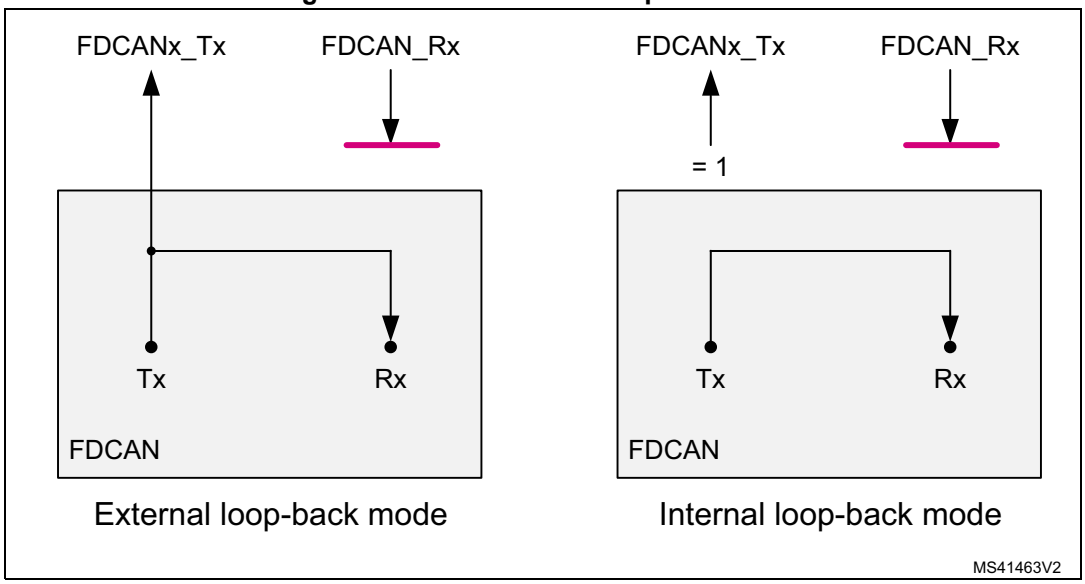
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 680* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 680* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 680. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

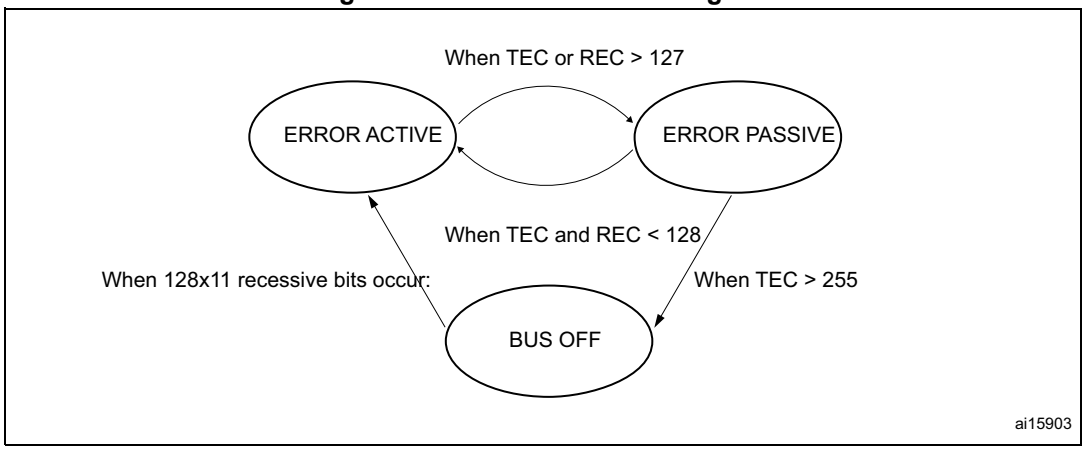
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

51.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 681. CAN error state diagram

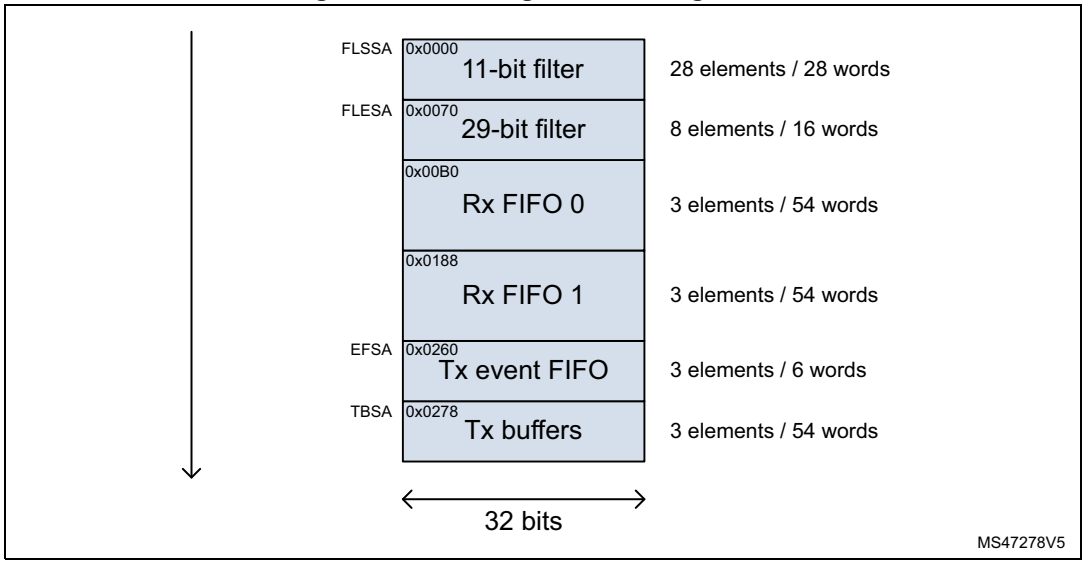


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

51.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 682*.

Figure 682. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 682*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

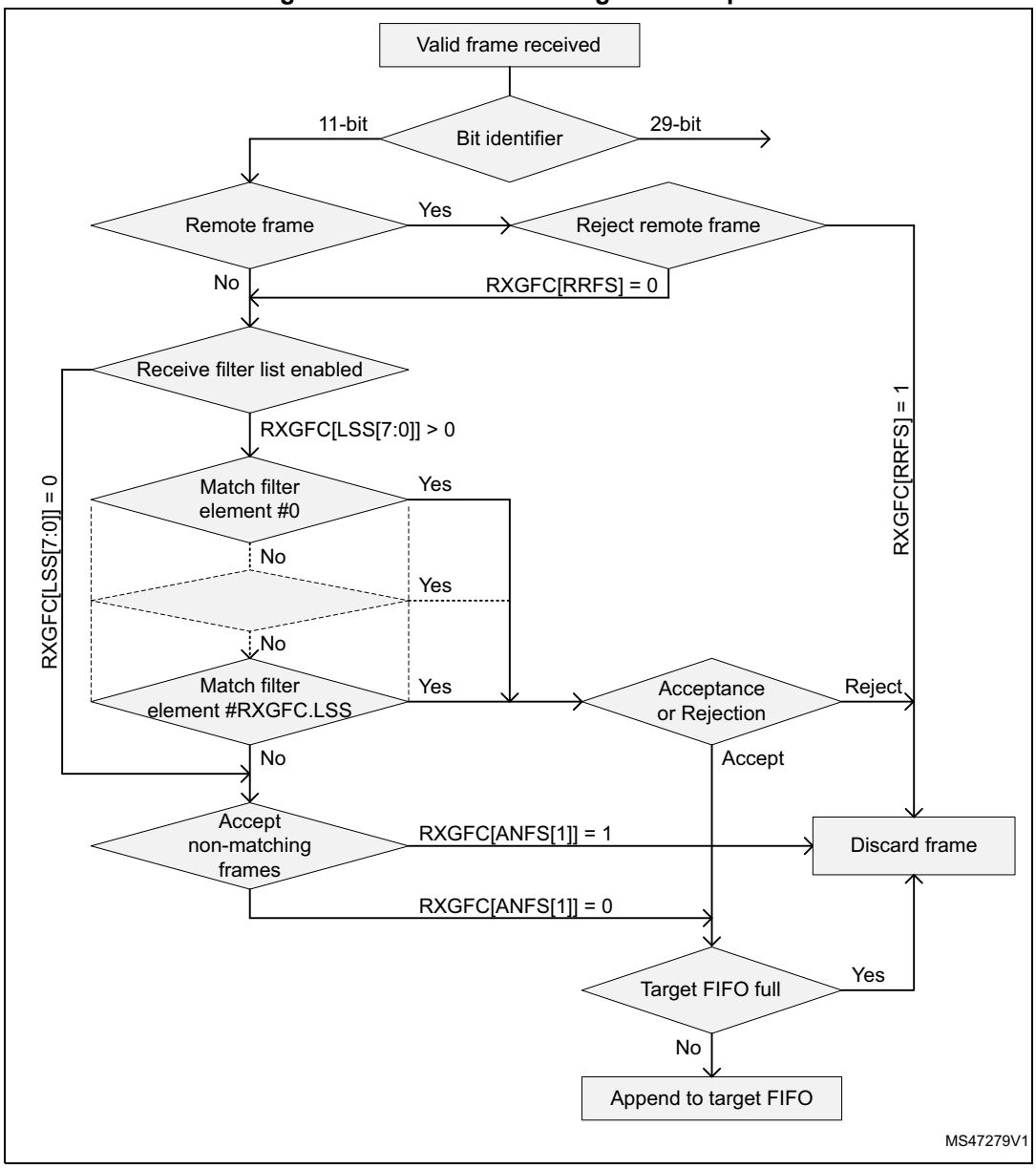
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 683 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 51.3.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 683. Standard message ID filter path

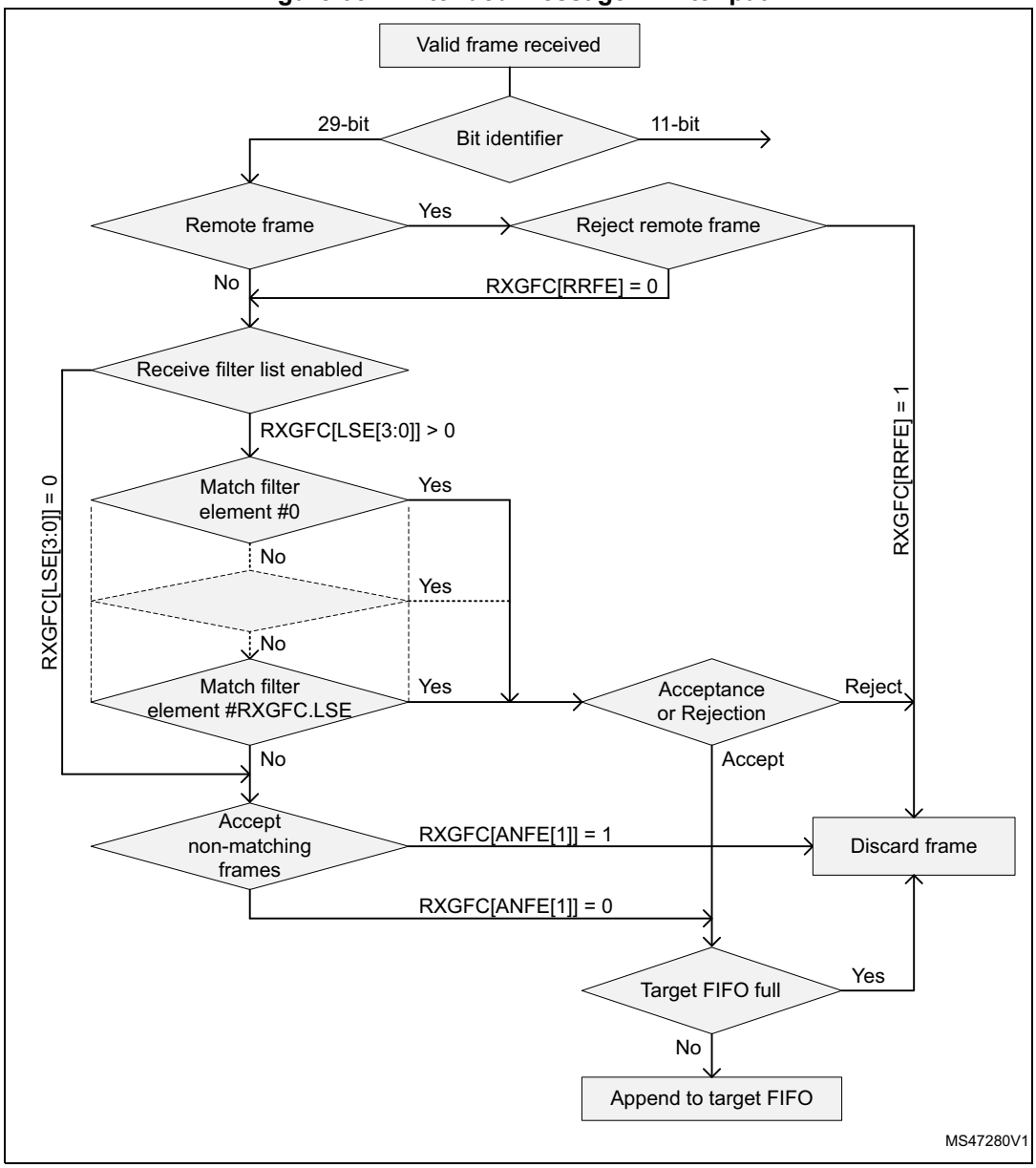


Extended message ID filtering

Figure 684 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 51.3.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 684. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 51.3.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 537. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

51.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 51.4.23* and *Section 51.4.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

51.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 538*. The description is provided in *Table 539*.

Table 538. Rx FIFO element

Bit	31			24	23							16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]												
R1	ANMF	FIDX[6:0]			Res.			FDF	BRS	DLC[3:0]			RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]								DB1[7:0]		D[7:0]	
R3	DB7[7:0]				DB6[7:0]								DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮								⋮			
Rn	DBm[7:0]				DBm-1[7:0]								DBm-2[7:0]		DBm-3[7:0]	

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 539. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 539. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 539. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

51.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 540. Tx buffer and FIFO element

Bit	31			24	23					16	15	8	7	0
T0	ESI	XTD	RTR	ID[28:0]										
T1	MM[7:0]				EFC	Res.	FDF	BRS	DLC[3:0]		Res.			
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]		D[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮						⋮			
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

Table 541. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 541. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 541. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

51.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 542. Tx event FIFO element

Bit	31			24	23						16	15		8	7		0
E0	ESI	XTD	RTR	ID[28:0]													
E1	MM[7:0]				ET[1:0]		EDL	BRS	DLC[3:0]			TXTS[15:0]					

Table 543. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 543. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

51.3.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 544. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 545. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

51.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 546. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 547. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

51.4 FDCAN registers

51.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

51.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value
The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

51.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods:
t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

51.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

51.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

51.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

51.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 48 MHz, the reset value of 0x0600 0A03 configures the FDCAN for a bit rate of 3 Mbit/s.

51.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3_cnt[0:15])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

51.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

51.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

51.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

51.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

51.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

51.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

51.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

51.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

51.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line has to be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
ARAL: Access to reserved address line
PEDL: Protocol error in data phase line
PEAL: Protocol error in arbitration phase line
WDIL: Watchdog interrupt line
BOL: Bus-off status
EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
EPL Error passive interrupt line
ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
TOOL: Timeout occurred interrupt line
MRAFL: Message RAM access failure interrupt line
TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
TEFLL: Tx event FIFO element lost interrupt line
TEFFL: Tx event FIFO full interrupt line
TEFNL: Tx event FIFO new entry interrupt line
TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
TCFL: Transmission cancellation finished interrupt line
TCL: Transmission completed interrupt line
HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
RF1LL: Rx FIFO 1 message lost interrupt line
RF1FL: Rx FIFO 1 full interrupt line
RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

51.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

51.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 683* and *Figure 684*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

51.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

51.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
r			r	r	r	r	r	r	r				r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

51.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

51.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

51.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

51.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

51.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:0 Reserved, must be kept at reset value.

51.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

51.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

51.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

51.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

51.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

51.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF[2:0]		
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

51.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

51.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.
Each Tx buffer has its own CFIE bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

51.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index
Tx event FIFO write index pointer.
Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

51.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

51.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

51.4.38 FDCAN register map

Table 548. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 548. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0

Table 548. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]					MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
	Reset value							0	0							0	0							0	0					0	0	0	0
0x009C	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN_TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0

Table 548. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																												0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

52 Universal serial bus full-speed host/device interface (USB)

52.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

52.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

52.3 USB implementation

Table 549 describes the USB implementation in the devices.

Table 549. STM32U3 USB implementation

USB features(1)	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

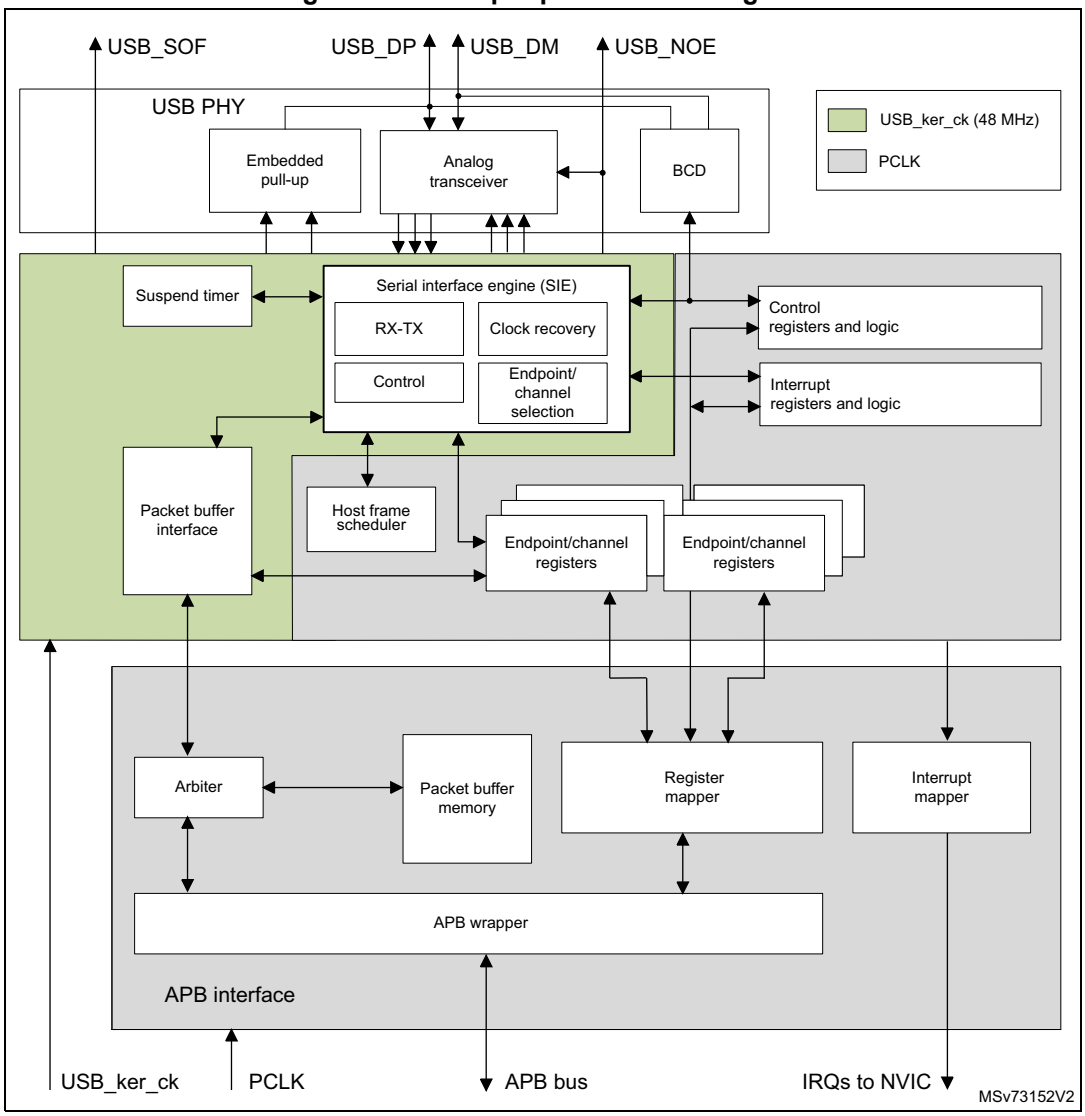
1. X= supported

52.4 USB functional description

52.4.1 USB block diagram

Figure 685 shows the block diagram of the USB peripheral.

Figure 685. USB peripheral block diagram



52.4.2 USB pins and internal signals

Table 550. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line

Table 550. USB input/output pins (continued)

Pin name	Pin type	Description
USB_NOE	Digital output	NOE (output enable of data lines)
USB_SOF	Digital output	SOF (start of frame indicator)

52.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

52.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

For LPM (link power management) this feature is not supported in Host mode.

52.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint

related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB2 bus through an APB2 interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB2 transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB2.
- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

52.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

52.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

52.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

52.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request

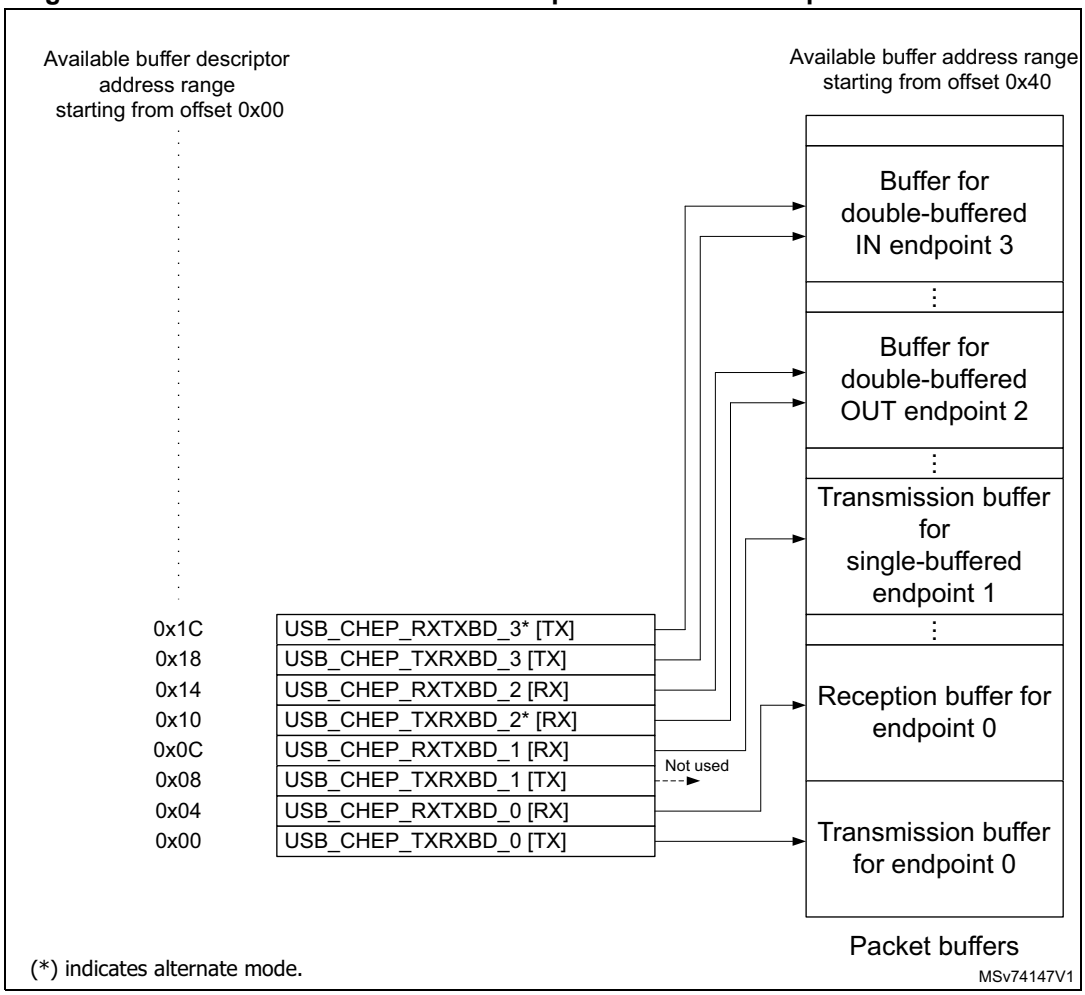
and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB2 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

Note: Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 16 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in *Section 52.7: USBSRAM registers*. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 52.5.5: Isochronous transfers in Device mode* and *Section 52.5.3: Double-buffered endpoints and usage in Device mode* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 686*.

For Host mode different sections explain the buffer usage model, notably *Section 52.5.6: Isochronous transfers in Host mode* and *Section 52.5.4: Double buffered channels: usage in Host mode*.

Figure 686. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can

change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to *Structure and usage of packet buffers on page 2305*) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the

indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USB_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer

description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal

OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

52.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 551. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 552. Bulk double-buffering memory buffers usage (Device mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 552*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

52.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below *Table 553: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 553. Bulk double-buffering memory buffers usage (Host mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

52.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to *Table 554*.

Table 554. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

52.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See *Table 555* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 555. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

52.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to ensure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 556*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 556. Resume event detection

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in *Table 557*, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 557. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Not allowed (noise on bus)	Go back in Suspend mode
“10”	Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume	None
“01”	Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

52.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB_CNTR
  - USB_ISTR
  - USB_FNR
  - USB_DADDR
  - USB_LPMCSR
  - USB_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB_CHEPnR

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

52.6.1 USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M
rw														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	r	rw	rw

Bit 31 **HOST:** HOST mode

HOST bit selects betweens host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

Bit 15 **CTRM:** Correct transfer interrupt mask
0: Correct transfer (CTR) interrupt disabled.
1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask
0: PMAOVR interrupt disabled.
1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask
0: ERR interrupt disabled.
1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wake-up interrupt mask
0: WKUP interrupt disabled.
1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask
0: Suspend mode request (SUSP) interrupt disabled.
1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
0: RESET interrupt disabled.
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask
0: SOF interrupt disabled.
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask
0: Expected start of frame (ESOF) interrupt disabled.
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask
0: LPM L1 state request (L1REQ) interrupt disabled.
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RES:** L1 remote wake-up / resume driver
– Device mode
Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
0: No effect
1: Send 50 µs remote-wake-up signaling to host

Bit 4 **L2RES:** L2 remote wake-up / resume driver

– Device mode

The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.

– Host mode

Software sets this bit to send resume signaling to the device.

Software clears this bit to send end of resume to device and restart SOF generation.

In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.

0: No effect

1: Send L2 resume signaling to device

Bit 3 **SUSPEN:** Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 **SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit power down
1: Enter power down mode

Bit 0 **USBRST:** USB Reset

– Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB.The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.
0: No effect
1: USB core is under reset

– Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.
0: No effect
1: USB reset driven

52.6.2 USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_ DCON	DCON_ STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR 512
	r	r												rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RST_ DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

– Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP:** Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame

– Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

52.6.3 USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

52.6.4 USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

52.6.5 USB LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]:** BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK:** LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.
1: the valid LPM token is ACK.
The NYET/ACK is returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

52.6.6 USB battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
rw								r	r	r	r	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET:** Data contact detection (DCD) status

– Device mode

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

Bit 3 **SDEN:** Secondary detection (SD) mode enable

– Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable

– Device mode

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 1 **DCDEN:** Data contact detection (DCD) mode enable

– Device mode

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable

– Device mode

This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

52.6.7 USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where *n* is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOG RX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOG TX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]:** Three errors for an IN transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]:** Three errors for an OUT or SETUP transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX:** Received error for an IN transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX:** Received error for an OUT/SETUP transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP:** Low speed endpoint – host with HUB only

– Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.
0: Full speed endpoint
1: Low speed endpoint

Bit 23 **NAK:**

– Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]:**

– Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX:** USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit
If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.
If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.
Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 **DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).
If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 52.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to *Section 52.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in *Table 558: Reception status encoding on page 2338*. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.
Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 52.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.
These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE
  DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.
- VALID
  A host channel is actively trying to submit USB transaction to device only when in VALID state.VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.
  VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.
- NAK
  NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE
- STALL
  STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

– Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

– Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 559: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.
The usage of isochronous channels/endpoints is explained in *Section 52.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 560* summarizes the different meanings.
DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 52.5.3: Double-buffered endpoints and usage in Device mode*.
STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

– Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written.

– Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 **DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).
If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 52.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to *Section 52.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX[1:0]:** Status bits, for transmission transfers

– Device mode

These bits contain the information about the endpoint status, listed in *Table 561*. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.
Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 52.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

The STATTX bits contain the information about the channel status. Refer to *Table 561* for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA[3:0]:** endpoint/channel address

– Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

– Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 558. Reception status encoding

STATRX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	**NAK**: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	**VALID**: this endpoint/channel is enabled for reception.

Table 559. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 560. Endpoint/channel kind meaning

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

Table 561. Transmission status encoding

STATTX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 561. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	**NAK**: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	**VALID**: this endpoint/channel is enabled for transmission.

52.6.8 USB register map

The table below provides the USB register map and reset values.

Table 562. USB register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_CHEP0R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USB_CHEP1R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**USB_CHEP2R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USB_CHEP3R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USB_CHEP4R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USB_CHEP5R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USB_CHEP6R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**USB_CHEP7R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 562. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20-0x3F	Reserved																																
0x40	**USB_CNTR**	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0														0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	**USB_ISTR**	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value		0	0												0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	**USB_FNR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	**USB_DADDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	**USB_LPMCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	**USB_BCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

52.7 USBSRAM registers

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 2305*.

52.7.1 Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

52.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 563*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 563. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	
16 (10000)	32 bytes	
...	...	...
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes

52.7.3 Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 563*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

52.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

52.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

Table 564. USBSRAM register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_ CHEP_ TXRXBD_0**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00	**USB_ CHEP_ TXRXBD_0 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	**USB_ CHEP_ RXTXBD_0**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	**USB_ CHEP_ RXTXBD_0 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	**USB_ CHEP_ TXRXBD_1**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	**USB_ CHEP_ TXRXBD_1 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	**USB_ CHEP_ RXTXBD_1**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	**USB_ CHEP_ RXTXBD_1 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
. . . .	. . . .	. . . .																															
0x38	**USB_ CHEP_ TXRXBD_7**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x38	**USB_ CHEP_ TXRXBD_7 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	**USB_ CHEP_ RXTXBD_7**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	**USB_ CHEP_ RXTXBD_7 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

53 Debug support (DBG)

53.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm CoreSight components.

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded Trace Macrocell™ (ETM)
- Cross trigger interface (CTI)
- Trace port interface unit (TPU)

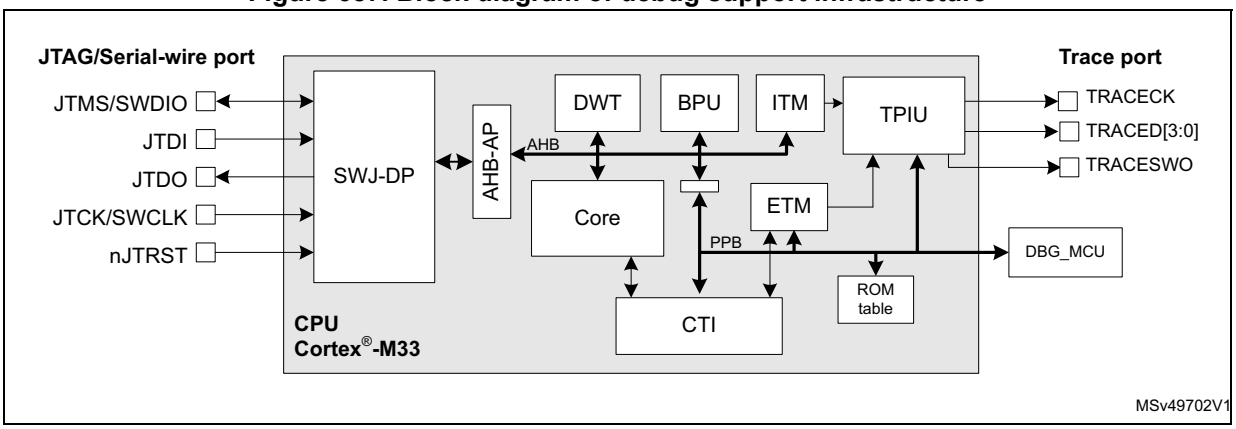
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm documents referenced in *Section 53.13: References*.

53.2 DBG functional description

53.2.1 DBG block diagram

Figure 687. Block diagram of debug support infrastructure



53.2.2 DBG pins and internal signals

Table 565. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial-wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	PA14
JTDI(1)	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
NJTRST	I	JTAG test reset	-	-	PB4

1. TDI is hosted on the same IO as a USBPD-CC line. To avoid pull-up/down conflict, a user option can help to decide whether the pad is used as TDI or as CC.

Table 566. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 567. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single-wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.

53.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain, that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the DP_CTRL_STAT. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in DP_CTRLSTAT. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

53.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in Shutdown or Standby low-power mode. To avoid losing the connection when the device enters Standby mode, the power can be maintained to the core by setting a bit in DBGMCU_CR. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

53.2.5 Debug and low-power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to *Section 53.12: Microcontroller debug unit (DBGMCU)*.

53.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features can not be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of nonsecure code.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features

  0: All debug features are disabled.

  1: Debug features in nonsecure state are enabled. Debug features in secure state are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in secure state when **dbgen** = 1.

  0: Debug features are disabled in secure state.

  1: Debug features are enabled in secure state.
- **niden**: enables trace and performance monitoring (non-invasive debug).

  0: Trace generation is disabled.

  1: Trace generation in nonsecure state is enabled. Trace generation in secure state is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in secure state when **niden** = 1.

  0: Trace generation is disabled in secure state.

  1: Trace generation is enabled in secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm technical documentation.

The state of the signals are set according to the readout protection (RDP) level (see *Section 3.11.1: Debug protection with readout protection (RDP)*), as shown in the table below:

Table 568. Authentication signal states

RDP level	Authentication signal state	Description
0	dbgen = 1, spiden = 1 niden = 1, spniden = 1	Debug and trace is enabled whatever the state of the processor. The debugger access to secure memory is permitted.
0.5	dbgen = 1, spiden = 0 niden = 1, spniden = 0	Debug and trace is enabled when the processor is in nonsecure state. The debugger access to secure memory is disabled.
1	dbgen = 1, spiden = 0 niden = 1, spniden = 0	Debug and trace is enabled when the processor is in nonsecure state. The debugger access to secure memory is disabled, as well as to the following areas: flash memory, SRAM2, backup registers, ICACHE, on-the-fly decryption region (OCTOSPI).
2	dbgen = 0, spiden = 0 niden = 0, spniden = 0	Debug and trace is disabled.

Note: Security features are only relevant when the option bit TZEN = 1. If security features are disabled, the authentication signals are still set according to the RDP level, but since the processor and all memories are nonsecure, spniden and spiden are redundant.

The state of the authentication signals can be read from DAUTHSTATUS register in the system control space (SCS) of the Cortex-M33.

The debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with PROT[6] set in AP_CSW.

The debugger access is disabled while the processor is booting from system flash memory (RSS), whatever the RDP level, if security features are enabled (TZEN = 1).

53.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) serial-wire debug port (SW-DP)

These two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

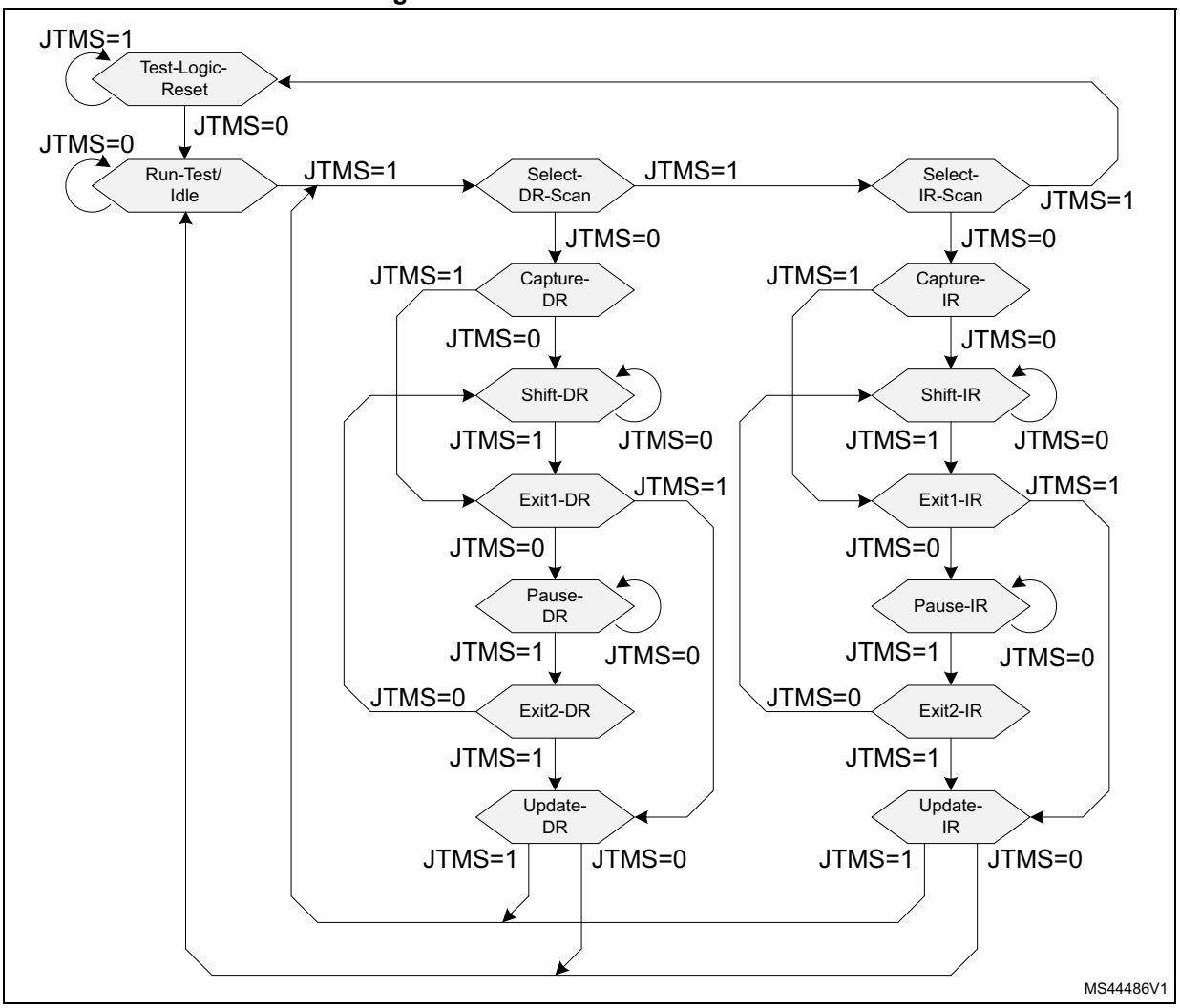
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

53.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 688. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred to the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   - The LSB of the IR scan chain is output on JTDO.
   - Bit[n] of the IR scan chain is transferred to bit[n-1].
   - The value on JTDI is transferred to the MSB of the IR scan chain.

3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register, and associated DR scan chain, are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred to the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occurs. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in the table below.

Table 569. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 31:1 = reserved – bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request bits 2:1 = A[3:2] = 2-bit address of a debug port register bit 0 = RnW = read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit acknowledge: – 010 = OK/FAULT – 001 = WAIT – others = reserved

Table 569. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request bits 2:1 = A[3:2] = 2-bit sub-address of an access port register bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – others= reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	Identification code 0x0BA0 4477: Cortex-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass: A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are detailed in the Arm Debug Interface Architecture Specification (see *Section 53.13: References*).

53.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host (see *Table 570*)
2. acknowledge response (3 bits) transmitted by the target (see *Table 571*)
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 572*)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 570. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Section 53.3.3: Debug port registers* – 1: AP register access - see *Section 53.4: Access ports*

Table 570. Packet request (continued)

Bit field	Name	Description
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 573* or *Table 574*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target.

Table 571. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 572. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case, the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

53.3.3 Debug port registers

Both SW-DP and JTAG-DP access the debug port (DP) registers listed in *Table 573*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 570*). The write data are sent in the data phase.
2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or

write to address 0x4 access the register corresponding to the contents of DP_SELECTR.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x0BE0 2477 (SW DP)

For JTAG_DP, reset value is 0x0BE0 1477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: Revision code
0x0: r0p0 (for JTAG_DP and SW_DP)

Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
0xBE

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0x1: MINDP implemented (transaction counter and pushed operations are not supported)

Bits 15:12 **VERSION[3:0]**: Debug port architecture version
0x1: DPv1 (JTAG_DP)
0x1: DPv2 (SW_DP)

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORT)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear
0: No effect
1: STICKYORUN bit cleared in DP_CTRL_STAT register

Bit 3 **WDERRCLR**: Write data error clear
0: No effect
1: WDATAERR bit cleared in DP_CTRL_STAT register

Bit 2 **STKERRCLR**: Sticky error clear
0: No effect
1: STICKYERR bit cleared in DP_CTRL_STAT register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: Current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: No effect
1: Transaction aborted

DP control and status register (DP_CTRL_STAT)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: Debug power-up acknowledge
See description in *Section 53.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: Debug power-up request
This bit controls the DCLK enable request signal.
0: Requests DCLK gating
1: Requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: Write data error (read-only) in SW-DP

This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to DP_ABORT.WDERRCLR.

0: No error

1: An error occurred

Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP

This bit indicates the response to the last AP read access.

0: Read not OK

1: Read OK

Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an error occurred in an AP transaction.It is reset by writing 1 to DP_ABORT.STKERRCLR (in SW-DP and JTAG-DP)

0: No error

1: An error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: Overrun (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. It is reset by writing 1 to DP_ABORT.ORUNERRCLR (in SW-DP and JTAG-DP).

0: No overrun

1: An overrun occurred

Bit 0 **ORUNDETECT**: Overrun detection mode enable

0: Disabled

1: Enabled. In the event of an overrun, the STICKYORUN bit is set, and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r	r						

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO

0x0: One data bit period

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode
0x0: Synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: Target revision
0x1: revision Z

Bits 27:12 **TPARTNO[15:0]**: Target part number
0x4550: STM32U375/385

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number
This bifield defines the instance number for the device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

DP event status register (DP_EVENTSTAT)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECT.DPBANKSEL[3:0] = 0x4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EA
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EA**: Event status flag
0: Cortex-M33 processor halted
1: Cortex-M33 processor not halted

DP event status register (DP_RESEND)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: Value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECT)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: Access port select
This bitfield selects the access port for the next transaction.
0x0: AP0 - Cortex-M33 debug access port (AHB-AP)
Others: Reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
This bitfield selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select
This bitfield selects the register at address 0x4 of the debug port.
0x0: DP_CTRL_STAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
0x4: DP_EVENTSTAT register
Others: Reserved

DP read buffer register (DP_RDBUFF)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Value returned by the last AP read access
The value returned by an AP read access can either be obtained using a second read access to the same address, that initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

53.3.4 Debug port register map

Table 573. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	DP_DPIDR	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	0	0	0	1	0	1	1	1	1	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	
0x0	DP_ABORT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0

Table 573. Debug port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4[1]	DP_CTRL_STAT	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x4[2]	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND [1:0]		WIREMODE [1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						
0x4[3]	DP_TARGETID	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x4[4]	DP_DLPIDR	TINSTANCE [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN [3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x4[5]	DP_EVENTSTAT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EA
	Reset value																																1
0x8	DP_RESEND	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	DP_SELECT	APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL [3:0]				DPBANKSEL [3:0]			
	Reset value	x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x
0xC	DP_RDBUFF	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.
5. DP_SELECTR.DPBANKSEL[3:0] = 0x4.

53.4 Access ports

There is one access port (AP) attached to the DP. It enables the access to the debug and trace features integrated in the Cortex-M33 processor core via its internal AHB bus.

53.4.1 Access port registers

The access port is of MEM-AP type: the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to

configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 574*.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of APBANKSEL[3:0] in DP_SELECT
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 569*), or of the SW-DP packet request (see *Table 570*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of DP_SELECT.APSEL[3:0] defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program APSEL[3:0] in DP_SELECT to choose the AP, and APBANKSEL[3:0] in DP_SELECT to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 570*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in AP_TAR.
2. Program AP_CSW, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from AP_DRW to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to AP_BDn triggers an access to TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP, refer to the Arm Debug Interface Architecture Specification (*Section 53.13: References*).

AP control/status word register (AP_CSW)

Address offset: 0x0

Reset value: 0x0100 00X0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT[6]	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
									r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PROT[6]**: Secure transfer request

This bit sets the protection attribute HPROT[6] of the bus transfer.

0: Secure transfer

1: Nonsecure transfer

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]**: Bus transfer protection
This bitfield sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX1: Data access (bit 24 is read only)
0bXX0X: Unprivilege mode
0bXX1X: Privilege mode
0bX0XX: Nonbufferable
0bX1XX: Bufferable
0b0XXX: Non-shareable, no look-up, non-modifiable
0b1XXX: Shareable, look-up, modifiable

Bits 23:7 Reserved, must be kept at reset value.

Bit 6 **DBGSTATUS**: Device enable (DEVICEEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address incremented by the size in bytes of the transaction (SIZE[2:0])
Other: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
Others: Reserved

AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]**: Address of current transfer

AP data read/write register (AP_DRWR)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]**: Data of current transfer

AP banked data n register (AP_BDn)

Address offset: 0x10 + 0x4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TBD[31:0]:** Banked data of current transfer to address TAR
TA + n × 4.
The auto address incrementing is not performed on AP_BD0-3. Banked transfers are only supported for word transfers.

AP configuration register (AP_CFG)

Address offset: 0xF4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LD**: Large data
0: Data not larger than 32 bits supported

Bit 1 **LA**: Long address
0: Physical addresses not larger than 32 bits supported

Bit 0 **BE**: Big endian
0: Only little-endian supported

AP base address register (AP_BASE)

Address offset: 0xF8

Reset value: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00FE

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base-address register format
1: Arm debug interface v5

Bit 0 **ENTRYPRESENT**: debug components presence
Indicates that debug components are present on the access port bus.
1: debug components present

AP identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x1477 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Revision number
0x1: r0p1

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bits 16:13 **CLASS[3:0]**:
0x8: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **VARIANT[3:0]**: Variant
0x1: Cortex-M33

Bits 3:0 **TYPE[3:0]**: Type
0x5: AHB5

53.4.2 Access port register map

Table 574. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP_CSW	Res.	PROT[6]	Res.	Res.	PROT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		0			0	0	0	1																		X	X	X		0	0	0
0x04	AP_TAR	TA[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	Reserved	Reserved																															
0x0C	AP_DRW	TD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x10 + 0x4 * n (n=0 to 3)	AP_BDn	TBD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF0	Reserved	Reserved																															
0xF4	AP_CFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
	Reset value																														0	0	0
0xF8	AP_BASE	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0											1	1
0xFC	AP_IDR	REVISION [3:0]				JEDECBANK [3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]		
	Reset value	0	0	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	1	0	1	0	1

53.5 ROM tables

53.5.1 MCU ROM table registers

MCU ROM memory type register (MCUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
0x1: System memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0054

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x54: STM32U375/385

MCU ROM CoreSight peripheral identity register 1(MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: STM32U375/385

MCU ROM CoreSight peripheral identity register 2 (MCUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

MCU ROM CoreSight component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.5.2 MCU ROM table register map

Table 575. MCU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	MCUROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	MCUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	MCUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	0	1	0	1	0	0
0xFE4	MCUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	MCUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	MCUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	MCUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	MCUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	MCUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	MCUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.5.3 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory
1: System memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC9: Cortex-M33

CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: Cortex-M33

CPU ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CPU ROM CoreSight component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CPU ROM CoreSight peripheral identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.5.4 Processor ROM table register map

Table 576. CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	CPUROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1

Table 576. CPU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFD0	CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	1
0xFE4	CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.6 Data watchpoint and trace unit (DWT)

53.6.1 DWT registers

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

DWT control register (DWT_CTRL)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT**: tTace sampling and exception tracing support (read only)
0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read only)
0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)
0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)
0: Supported

Bit 23 **CYCDISS**: Cycle counter disabled secure
Controls whether the cycle counter is disabled in secure mode.
0: No effect
1: Disable incrementing of the cycle counter when the processor is in secure state

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA**: Enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA**: Enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: Enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of the synchronization packet counter tap on the CYCCNT counter
This bitfield determines the synchronization packet rate.
00: Disabled, no synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter
Writes to this bitfield are ignored if the POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: Reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: Enable CYCCNT counter
0: Disabled
1: Enabled

DWT cycle count register (DWT_CYCCNT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: Processor clock-cycle counter

DWT CPI count register (DWT_CPICNT)

Address offset: 0x008

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter

Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNT)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter
Counts the number of cycles spent in Sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNT)

Address offset: 0x014

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter
Counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNT)

Address offset: 0x018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value.
Samples the current value of the program counter.

DWT comparator x register (DWT_COMPx)

Address offset: 0x020 + 0x10 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]**: Reference value for comparison

DWT function register 0 (DWT_FUNCTION0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle counter, instruction address, data address, and data address with value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
Indicates if a comparator match occurred since the register was last read.
0: No match
1: A match occurred.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size
Defines the size of the object being watched for by data value and data address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generates debug event

0x2: For a cycle counter, instruction address, data address, data value, or linked data value comparator, generate a data trace match packet. For a data address with value comparator, generate a data trace data value packet.

0x3: For a data address limit comparator, generate a data trace data address packet. For a cycle counter, instruction address limit, or data address comparator, generate a data trace PC value packet. For a data address with value comparator, generate both a data trace PC value packet and a data trace data value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 0.

For possible values of this bitfield, refer to *Section 53.13* [3].

DWT function register 1 (DWT_FUNCTION1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

Identifies the capability for match for comparator 1.

0b11010: instruction address, instruction address limit, data address, data address limit, and data address with value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

Indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by data value and data address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generates debug event

0x2: For a cycle counter, instruction address, data address, data value, or linked data value comparator, generate a data trace match packet. For a data address with value comparator, generate a data trace Data value packet.

0x3: For a data address limit comparator, generate a data trace data address packet. For a cycle counter, instruction address limit, or data address comparator, generate a data trace PC value packet. For a data address with value comparator, generate both a data trace PC value packet and a data trace data value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 1.

For possible values of this bitfield, refer to *Section 53.13* [3].

DWT function register 2 (DWT_FUNCTION2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

Identifies the capability for match for comparator 2

0b01010: instruction address, data address, and data address with value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

Indicates if a comparator match occurred since the register was last read.

0: No match

1: A match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by data value and data address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generates debug event

0x2: For a cycle counter, instruction address, data address, data value, or linked data value comparator, generate a data trace match packet. For a data address with value comparator, generate a data trace data value packet.

0x3: For a data address limit comparator, generate a data trace data address packet. For a cycle counter, instruction address limit, or data address comparator, generate a data trace PC value packet. For a data address with value comparator, generate both a data trace PC value packet and a data trace data value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 2.

For possible values of this bitfield, refer to *Section 53.13* [3]

DWT function register 3 (DWT_FUNCTION3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Capability identification

Identifies the capability for match for comparator 2.

0b11110: instruction address, instruction address limit, data address, data address limit, data value, linked data value, and data address with value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match

Indicates if a comparator match has occurred since the register was last read.

0: No match

1: A match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by data value and data address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: Trigger only

0x1: Generates debug event

0x2: For a cycle counter, instruction address, data address, data value, or linked data value comparator, generate a data trace match packet. For a data address with value comparator, generate a data trace data value packet.

0x3: For a data address limit comparator, generate a data trace data address packet. For a cycle counter, instruction address limit, or data address comparator, generate a data trace PC value packet. For a data address with value comparator, generate both a data trace PC value packet and a data trace data value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 2.

For possible values of this bitfield, refer to *Section 53.13* [3]

DWT device type architecture register (DWT_DEVARCH)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code

0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DWT_DEVARCH register present

0x1: present

Bits 19:16 **REVISION[3:0]**: Architecture revision

0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version

0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part

0xA02: DWT architecture

DWT device type register (DWT_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x0: Other

Bits 3:0 **MAJOR[3:0]**: Major type
0x0: Miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: Cortex-M33 DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: Cortex-M33 DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.6.2 DWT register map

Table 577. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRL	NUMCOMP [3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT [3:0]				POSTPRESET [3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0

Table 577. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DWT_CYCCNT	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x00C	DWT_EXCCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x010	DWT_SLPCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	DWT_LSUCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	DWT_FOLDCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x020	DWT_COMP0	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x024	Reserved	Reserved																															
0x028	DWT_FUNCTION0	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]		
	Reset value	0	1	0	1	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	DWT_COMP1	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x034	Reserved	Reserved																															
0x038	DWT_FUNCTION1	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]		
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	DWT_COMP2	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x044	Reserved	Reserved																															
0x048	DWT_FUNCTION2	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]		
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	DWT_COMP3	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x054	Reserved	Reserved																															

Table 577. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	DWT_FUNCTION3	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C-0xFC4	Reserved	Reserved																															
0xFC8	DWT_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0xFCC	DWT_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.7 Instrumentation trace macrocell (ITM)

53.7.1 ITM registers

ITM stimulus register x (ITM_STIMx)

Address offset: 0x000 + 0x4 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:0 **STIMULUS[31:0]**: Trace output data

When writing, write data is output on the trace bus as a software event packet.

When reading:

- Bit 1 is a disable flag:
  - 0: Stimulus port and ITM enabled
  - 1: Stimulus port and ITM disabled
- Bit 0 is a FIFO ready indicator:
  - 0: Stimulus port buffer is full (or port is disabled)
  - 1: Stimulus port can accept new write data

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable

Each bit x (0 to 31) enables the stimulus port associated with the ITM_STIMx register.

0: Port disabled

1: Port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: Disable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports.
XXX0: Unprivileged access permitted on ports 0 to 7
XXX1: Only privileged access permitted on ports 0 to 7
XX0X: Unprivileged access permitted on ports 8 to 15
XX1X: Only privileged access permitted on ports 8 to 15
X0XX: Unprivileged access permitted on ports 16 to 23
X1XX: Only privileged access permitted on ports 16 to 23
0XXX: Unprivileged access permitted on ports 24 to 31
1XXX: Only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: Busy bit
Indicates whether the ITM is currently processing events.
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this bitfield.
Note: Different identifiers must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescaler, used with the trace packet reference clock
0x0: No prescaling
0x1: Divides by 4.
0x2: Divides by 16.
0x3: Divides by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: Stall enable
0: Drops hardware source packets and generates an overflow if the ITM output is stalled.
1: Stalls the processor to guarantee delivery of data trace packets.

Bit 4 **SWOENA**: SWO enable
Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: Transmit enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRL.SYNCTAP field for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA**: Local timestamp generation enable
0: Disabled
1: Enabled

Bit 0 **ITMENA**: ITM enable
0: Disabled
1: Enabled

ITM device type architecture register (ITM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x4: Associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: Major type
0x3: Trace source

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.7.2 ITM register map

Table 578. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4 * x, (x=0 to 31)	ITM_STIMx	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x07C-0xDFC	Reserved	Reserved																															
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04-0xE3C	Reserved	Reserved																															
0xE40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44-0xE7C	Reserved	Reserved																															
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84-0xFB8	Reserved	Reserved																															
0xFBC	ITM_DEVARCH	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	ITM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 578. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.8 Breakpoint unit (BPU)

53.8.1 BPU registers

BPU control register (BPU_CTRL)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r					r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: Revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]**: Number of instruction address comparators supported
0x08: Eight instruction comparators supported

Bits 11:8, 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to BPU_CTRL register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: BPU enable
0: Disabled
1: Enabled

BPU comparator x register (BPU_COMPx)

Address offset: 0x008 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: Breakpoint address

Bit 0 **BE**: Breakpoint enable
0: Disabled
1: Enabled

BPU device type architecture register (BPU_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: BPU architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA03: BPU architecture

BPU device type register (BPU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Subtype
0x0: Other

Bits 3:0 **MAJOR[3:0]**: Major type
0x0: Miscellaneous

BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: BPU part number

BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.8.2 BPU register map

Table 579. BPU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BPU_CTRL	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved	Reserved																															

Table 579. BPU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008+ 0x4 * x, (x=0 to 7)	BPU_COMPx	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028- 0xFB8	Reserved	Reserved																															
0xFBC	BPU_DEVARCH	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0- 0xFC8	Reserved	Reserved																															
0xFCC	BPU_DEVTYPE	Res.																								SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	BPU_PIDR4	Res.																								SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4- 0xFDC	Reserved	Reserved																															
0xFE0	BPU_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	BPU_PIDR1	Res.																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.																								REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	BPU_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.																								CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	BPU_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	BPU_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.9 Embedded Trace Macrocell (ETM)

53.9.1 ETM registers

The ETM registers are located at address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace unit enable
0: Disabled
1: Enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Stability status
Indicates that the ETM-M33 registers are stable and can be read.
0: Not stable
1: Stable

Bit 0 **IDLE**: Trace unit status
Indicates that the trace unit is inactive.
0: Not idle
1: Idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Returns stack enable
- 0: Disabled
- 1: Enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: Conditional instruction tracing
- 0x0: Conditional instruction tracing disabled
- 0x1: Conditional load instructions traced
- 0x2: Conditional store instructions traced
- 0x3: Conditional load and store instructions traced
- 0x7: All conditional instructions traced
- Others: Reserved

Bit 4 **CCI**: Cycle counting in instruction trace
- 0: Disabled
- 1: Enabled

Bit 3 **BB**: Branch broadcast mode
- 0: Disabled
- 1: Enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event1
0: Single selected resource
1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource number based on TYPE1
Selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: Resource type for event0
0: Single selected resource
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: Resource number based on TYPE0
Selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low-power state behavior override
0: Normal low-power state behavior
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: Instruction event generation
Enables generation of an event element in the instruction stream.
00: Neither event0 nor event1 causes an event element.
01: Event0 causes an event element when it occurs. Event1 does not cause an event element.
10: Event0 does not cause an event element. Event1 causes an event element when it occurs
11: Both event0 and event1 cause an event element when they occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: Instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: Processor stalling
Stalls processor based on instruction trace buffer space.
0: The ETM must not stall the processor.
1: The ETM can stall the processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active
This bitfield provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.
0x0: Zero invasion, but greater risk of FIFO overflow.
...
0xF: Maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Synchronization period

Defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.
0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Instruction trace cycle-count threshold

Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This bitfield must be programmed with a unique value to differentiate it from other trace sources in the system.
Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Zxception level in secure state

Controls whether instruction tracing is enabled for the corresponding exception level, in secure state.

0bXXX0: Instruction trace not generated in secure state, for exception level 0

0bXXX1: Instruction trace generated in secure state, for exception level 0

0b0XXX: Instruction trace not generated in secure state, for exception level 3

0b1XXX: Instruction trace generated in secure state, for exception level 3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Trace system error exception

0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.

1: The system error exception is always traced.

Bit 10 **TRCRESET**: Trace reset exception

0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

1: The reset exception is always traced.

Bit 9 **SSSTATUS**: Start/stop logic status

0: Stopped

1: Started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: Event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value
This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth
Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used
0x0: no P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: Number of P1 right-hand keys used (including normal and special keys)
0x0: no P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Number of special P1 right-hand keys used
0x0: No special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**: Number of conditional instruction right-hand keys used (including normal and special keys)
0x1: One conditional instruction right-hand key implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys used
0x0: No special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: Implementation specific extension support
0x0: No implementation specific extensions supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Commit field meaning
Indicates the meaning of the commit field in some packets.
1: Commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: Trace data transfers for exceptions
Indicates support for the tracing of data transfers for exceptions and exception returns.
0: Not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support
0: Not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Conditional result tracing
Indicates how conditional results are traced.
0: The trace unit indicates only if a conditional instruction passes or fails its condition code check.

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported
0x1: Two events

Bit 9 **RETSTACK**: Return stack support
1: Two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Cycle counting support
1: Cycle counting implemented

Bit 6 **TRCCOND**: Conditional instruction support
1: Conditional instruction tracing implemented

Bit 5 **TRCBB**: Branch broadcast support
1: Branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: Data tracing support
0x0: Data tracing not supported

Bits 2:1 **INSTP0[1:0]**: Support for tracing of load and store instructions as P0 elements
0x0: Not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer
0x41: Arm

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x2: Minor revision 2

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x1: Implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size
0x0: Data value size not supported

Bits 19:15 **DASIZE[4:0]**: Data address size
0x0: Data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: Maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: Number of processors available for tracing
0x0: One processor

Bit 27 **SYSSTALL**: System support for stall control of the processor
1: System supports stall control

Bit 26 **STALLCTL**: Stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: Trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: Implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Privilege level implementation
0x9: Privilege level thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Minimum value that can be programmed to ETM_CCCTLR.THRESHOLD
Defines the minimum cycle counting threshold.
0x4: Minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: Number of context ID comparators
0x0: Context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls
0x1: One single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs
0x1: Two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs for the DWT
0x4: Four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: Data address comparisons
0: Data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators
0x0: No data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs
0x0: No address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Reduced function counter
1: Counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: Number of counters
0x1: One counter implemented

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states
0x0: No sequencer states implemented

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Low-power state override support
1: Low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of external input selectors
0x0: No external input selectors implemented

Bits 8:0 **NUMEXTIN[8:0]**: Number of external inputs
0x004: Four external inputs implemented

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Result of a combined pair of resources inversion
0: Not inverted
1: Inverted

Bit 20 **INV**: Selected resource inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resource selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparator element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resource selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Selected resource inversion
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Group of resource selection
0x0: External input selectors (select 0-3)
0x1: Inputs from processor DWT comparators element (select 0-3)
0x2: Counter at zero (select 0)
0x3: Single-shot comparator (select 0)
Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: More resource selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: Reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: Single-shot comparator status
Indicates whether any of the selected comparators have matched.
0: No match occurred
1: At least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: Processor comparator input sensitivity
1: Single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: Data value comparator support
0: Single-shot data value comparisons not supported

Bit 1 **DA**: data address comparator support
0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support
0: Single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: Processor comparator inputs selection for single-shot control
0xXXX0: Processor comparator input 0 not selected
0xXXX1: Processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: Processor comparator input 1 selected
0xX0XX: Processor comparator input 2 not selected
0xX1XX: Processor comparator input 2 selected
0x0XXX: Processor comparator input 3 not selected
0x1XXX: Processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power-up request
0: Power-up not requested
1: Power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

ETM claim tag set register (ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bits setting
Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are 4 bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bit reset
Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x2: Secure noninvasive debug disabled
0x3: Secure noninvasive debug enabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x2: Nonsecure noninvasive debug disabled
0x3: Nonsecure noninvasive debug enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

ETM device type architecture register (ETM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: Architecture part
0xA13: ETM architecture

ETM CoreSight device type register (ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device subtype identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ETM CoreSight peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.9.2 ETM register map

Table 580. ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETM_PRGCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved	Reserved																															
0x00C	ETM_STATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															X	X
0x010	ETM_CONFIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014-0x01C	Reserved	Reserved																															
0x020	ETM_EVENTCTL0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	ETM_EVENTCTL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved	Reserved																															

Table 580. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	ETM_STALLCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved	Reserved																															
0x034	ETM_SYNCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]			
	Reset value																													0	1	0	1	0
0x038	ETM_CCCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved	Reserved																															
0x040	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044-0x07C	Reserved	Reserved																															
0x080	ETM_VICTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084-0x13C	Reserved	Reserved																															
0x140	ETM_CNTRLDVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144-0x17C	Reserved	Reserved																															
0x180	ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198-0x1BC	Reserved	Reserved																															
0x1C0	ETM_IMSPECR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
	Reset value																													0	0	0	0
0x1C4-0x1DC	Reserved	Reserved																															

Table 580. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]			REVISION [3:0]				
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]			CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	ETM_IDR4	NUMVMIDC [3:0]				NUMCIDC [3:0]				NUMSSCC [3:0]				NUMRSPAIR [3:0]				NUMPC [3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE [5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8-0x204	Reserved	Reserved																															
0x208	ETM_RSCTLR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	ETM_RSCTLR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210-0x27C	Reserved	Reserved																															
0x280	ETM_SSCCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								X																								
0x284-0x29C	Reserved	Reserved																															
0x2A0	ETM_SSCSR0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4-0x2BC	Reserved	Reserved																															

Table 580. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C0	ETM_SSPCICR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4-0x30C	Reserved	Reserved																															
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318-0xF9C	Reserved	Reserved																															
0xFA0	ETM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFB4	Reserved	Reserved																															
0xFB8	ETM_AUTHSTAT	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	ETM_DEVARCH	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	ETM_DEVTYPE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	ETM_PIDR0	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 580. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.10 Trace port interface unit (TPIU)

53.10.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Trace port sizes, from 1 to 32 pins
Bit n-1 when set, indicates that port size n is supported.
0x0000 000F: Port sizes 1 to 4 supported

TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: urrent trace port size

Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result.

This register must only be modified when the formatter is stopped.

TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: Baudrate for the asynchronous output TRACESWO

The baudrate is given by the TRACELKIN frequency divided by (PRESCALER + 1).

TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: Protocol used for trace output

0x0: Parallel trace port mode

0x1: Asynchronous SWO using Manchester encoding

0x2: Asynchronous SWO using NRZ encoding

0x3: Reserved

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Formatter stop

Indicates whether the formatter can be stopped or not.

1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stop

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: The formatter has not stopped.

Bit 0 **FLINPROG**: Flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: No flush in progress

1: Flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: Trigger on trigger in

1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Flush on manual
0: Flush completed
1: Generates a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Continuous formatting enable
Setting this bit to zero in SWO mode bypasses the formatter, and only ITM/DWT trace is output. ETM trace is discarded.
0: Continuous formatting disabled
1: Continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]**: Formatter frame counter
Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bit setting
Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are 4 bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bit reset
Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support
0x1: Supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: Supported

Bit 9 **TCLKDATA**: Trace clock plus data support
0x0: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of two
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x1: Two input ports

TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Subclassification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: common identification value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.10.2 TPIU register map

Table 581. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008	Reserved	Reserved																															
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0EC	Reserved	Reserved																															
0x0F0	TPIU_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
	Reset value																															0	1
0x0F4 to 0x2FC	Reserved	Reserved																															
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	

Table 581. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	TPIU_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
030-0xF9C	Reserved	Reserved																															
0xFA0	TPIU_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0FA8-0xFC4	Reserved	Reserved																															
0xFC8	TPIU_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
0xFCC	TPIU_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.11 Cross-trigger interface (CTI)

53.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

CTI control register (CTI_CONTROL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global CTI enable

0: Disabled

1: Enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: Channel event setting
Read:
XXX0: Channel 0 event inactive
XXX0: Channel 0 event active
XX0X: Channel 1 event inactive
XX1X: Channel 1 event active
X0XX: Channel 2 event inactive
X1XX: Channel 2 event active
0XXX: Channel 3 event inactive
1XXX: Channel 3 event active
Write:
XXX0: No effect
XXX0: Sets event on channel 0.
XX0X: No effect
XX1X: Sets event on channel 1.
X0XX: No effect
X1XX: Sets event on channel 2.
0XXX: No effect
1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEA)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: Channel event clear
0000: No effect
XXX1: Clears event on channel 0.
XX1X: Clears event on channel 1.
X1XX: Clears event on channel 2.
1XXX: Clears event on channel 3.

CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: Pulse channel event
This register clears itself immediately.
0000: No effect
XXX1: Generates pulse on channel 0.
XX1X: Generates pulse on channel 1.
X1XX: Generates pulse on channel 2.
1XXX: Generates pulse on channel 3.

CTI trigger input x enable register (CTI_INENx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: Trigger input event enable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on channels.

XXX1: Trigger x generates events on channel 0.

XX1X: Trigger x generates events on channel 1.

X1XX: Trigger x generates events on channel 2.

1XXX: Trigger x generates events on channel 3.

CTI trigger output x enable register (CTI_OUTENx)

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: Trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger input status register (CTI_TRIGINSTATUS)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status

There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger output status register (CTI_TRIGOUTSTATUS)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel input status register (CTI_CHINSTATUS)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTATUS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: Channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATE)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: Channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.
0000: Channel events do not propagate.
XXX1: Channel 0 events propagate.
XX1X: Channel 1 events propagate.
X1XX: Channel 2 events propagate.
1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available

0x4: Four channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available

0x8: Eight trigger inputs and eight trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Subclassification
0x1: Cross-triggering component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x21: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0xD: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.11.2 CTI register map

Table 582. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004-0x00C	Reserved	Reserved																															
0x010	CTI_INTACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	CTI_APPSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													X	X	X	X
0x020 + 0x4 * x, (x=0 to 7)	CTI_INENx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040-0x09C	Reserved	Reserved																															
0x0A0 + 0x4 * x, (x=0 to 7)	CTI_OUTENx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0-0x12C	Reserved	Reserved																															
0x130	CTI_TRIGINSTATUS	Res	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRIGOUTSTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS [3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
	Reset value																													0	0	0	0

Table 582. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	CTI_GATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144-0xFC4	Reserved	Reserved																															
0xFC8	CTI_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	0	1
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.12 Microcontroller debug unit (DBGMCU)

53.12.1 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the AHB access port, and to software, at base address 0xE004 4000.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0xXXXX 6XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
0x1001: Revision Z

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identification
0x454: STM32U375/385

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
								rw	rw	rw	rw		rw	rw	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]**: Trace pin assignment

0x0: Trace pins assigned for asynchronous mode (TRACESWO)

0x1: Trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)

0x2: Trace pins assigned for synchronous mode with a port width of 2 (TRACECK, TRACED0-1)

0x3: Trace pins assigned for synchronous mode with a port width of 4 (TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: Trace port and clock enable.

This bit enables the trace port clock, TRACECK.

0: Disabled

1: Enabled

Bit 4 **TRACE_IOEN**: Trace pin enable

0: Disabled - trace pins not assigned

1: Enabled - trace pins assigned according to the value of TRACE_MODE bitfield

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allows debug in Standby mode

0: Normal operation. All clocks are disabled and the core powered down automatically in Standby mode.

1: Automatic clock stop/power-down disabled. All active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Allows debug in Stop mode

0: Normal operation. All clocks are disabled automatically in Stop mode.

1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
	rw							rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw						rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_RTC_STOP**: RTC stop in debug

0: Normal operation. The RTC continues to operate while the CPU is in debug mode.

1: Stop in debug. The RTC is frozen while the CPU is in debug mode.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I3C1_STOP**: I3C1 timeout stop in debug
0: Normal operation. I3C1 timeout continues to operate while the CPU is in debug mode.
1: Stop in debug. I3C1 timeout is frozen while the CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
0: Normal operation. I2C2 SMBUS timeout continues to operate while the CPU is in debug mode.
1: Stop in debug. I2C2 SMBUS timeout is frozen while the CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while the CPU is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while the CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in debug
0: Normal operation. The IWDG continues to operate while the CPU is in debug mode.
1: Stop in debug. The IWDG is frozen while the CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug
0: Normal operation. The WWDG continues to operate while CPU is in debug mode.
1: Stop in debug. The WWDG is frozen while CPU is in debug mode.

Bits 10:6 Reserved, must be kept at reset value.

Bits 5:4 **DBG_TIMy_STOP**: TIMy stop in debug (y = 7 to 6)
0: Normal operation. The TIMy continues to operate while the CPU is in debug mode.
1: Stop in debug. The TIMy is frozen while the CPU is in debug mode.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DBG_TIMy_STOP**: TIMy stop in debug (y = 4 to 2)
0: Normal operation. The TIMy continues to operate while the CPU is in debug mode.
1: Stop in debug. The TIMy is frozen while the CPU is in debug mode.

DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: Normal operation. The LPTIM2 continues to operate while the CPU is in debug mode.
1: Stop in debug. The LPTIM2 is frozen while the CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DBG_I3C2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
				rw									rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **DBG_I3C2_STOP**: I3C2 timeout stop in debug
0: Normal operation. I3C2 timeout continues to operate while the CPU is in debug mode.
1: Normal in debug. I3C2 timeout is frozen while the CPU is in debug mode.

Bits 26:19 Reserved, must be kept at reset value.

Bits 18:16 **DBG_TIMy_STOP**: TIMy stop in debug (y = 17 to 15)
0: Normal operation. The TIMy continues to operate while the CPU is in debug mode.
1: Stop in debug. The TIMy is frozen while the CPU is in debug mode.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: Normal operation. The TIM1 continues to operate while the CPU is in debug mode.
1: Stop in debug. The TIM1 is frozen while the CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.
												rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw										

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug
0: Normal operation. The LPTIM4 continues to operate while the CPU is in debug mode.
1: Stop in debug. The LPTIM4 is frozen while the CPU is in debug mode.

Bit 18 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug
0: Normal operation. The LPTIM3 continues to operate while the CPU is in debug mode.
1: Stop in debug. The LPTIM3 is frozen while the CPU is in debug mode.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: Normal operation. The LPTIM1 continues to operate while the CPU is in debug mode.
1: Stop in debug. The LPTIM1 is frozen while the CPU is in debug mode.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **DBG_I2C3_STOP**: I2C3 stop in debug
0: Normal operation. The I2C3 continues to operate while the CPU is in debug mode.
1: Stop in debug. The I2C3 is frozen while the CPU is in debug mode.

Bits 9:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_GPDMA11_STOP	DBG_GPDMA10_STOP	DBG_GPDMA9_STOP	DBG_GPDMA8_STOP	DBG_GPDMA7_STOP	DBG_GPDMA6_STOP	DBG_GPDMA5_STOP	DBG_GPDMA4_STOP	DBG_GPDMA3_STOP	DBG_GPDMA2_STOP	DBG_GPDMA1_STOP	DBG_GPDMA0_STOP
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DBG_GPDMAy_STOP**: GPDMA channel y stop in debug (y = 11 to 0)
0: normal operation. GPDMA channel y continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel y is frozen while the CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0xXXXX 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_PRESENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug host authentication register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTH_KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **AUTH_KEY[31:0]**: Device authentication key
The device specific 128-bit authentication key (OEM key) must be written to this register (in four successive 32-bit writes, least significant word first) to permit RDP regression. Writing a wrong key locks access to the device, and prevent code execution from the flash memory.

DBGMCU debug device authentication register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUTH_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTH_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUTH_ID[31:0]**: Device specific authentication ID
Device specific authentication ID used for RDP regression.

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

53.12.2 DBGMCU register map

Table 583. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE[1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																									0	0	0	0		0	0	
0x008	DBGMCU_APB1LFZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value		0							0	0	0									0	0					0	0		0	0	0
0x0C0	DBGMCU_APB1HFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					

Table 583. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	DBG_I3C2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0									0	0	0					0											
0x010	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	DBG_I3C2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0									0	0	0					0											
0x04	DBGMCU_APB3FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0							0										
0x018-0x01C	Reserved	Reserved																															
0x020	DBGMCU_AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA11_STOP	DBG_GPDMA10_STOP	DBG_GPDMA9_STOP	DBG_GPDMA8_STOP	DBG_GPDMA7_STOP	DBG_GPDMA6_STOP	DBG_GPDMA5_STOP	DBG_GPDMA4_STOP	DBG_GPDMA3_STOP	DBG_GPDMA2_STOP	DBG_GPDMA1_STOP	DBG_GPDMA0_STOP
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x0F8	Reserved	Reserved																															
0x0FC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x100	DBGMCU_DBG_AUTH_HOST	AUTH_KEY[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x104	DBGMCU_DBG_AUTH_DEVICE	AUTH_ID[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x108-0xFBC	Reserved	Reserved																															
0xFD0	DBGMCU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	DBGMCU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 583. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	DBGMCU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE8	DBGMCU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	DBGMCU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DBGMCU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DBGMCU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	DBGMCU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DBGMCU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

53.13 References

1. IHI 0031C (ID080813) - Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm CoreSight SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015DDI 0314H - Arm CoreSight Components Technical Reference Manual, Issue H, 10 July, 2009
3. DDI 0553A (ID092917) - Arm v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
4. 100230_0002_00_en - Arm Cortex-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
5. 100232_0001_00_en - Arm CoreSight ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

54 Device electronic signature

The device electronic signature is stored in the system memory area of the flash memory module and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the devices.

54.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal flash memory
- to activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x0BFA 0700

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

54.2 Flash memory size data register

Base address: 0x0BFA 07A0

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This bitfield indicates the size of the device flash memory expressed in Kbytes.
As an example, 0x400 corresponds to 1024 Kbytes.

54.3 Package data register

Base address: 0x0BFA 0500

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
											r	r	r	r	r

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type

00110: LQFP32
00101: LQFP48/UQFN48
00000: LQFP64/UFBGA64
00010: LQFP100/UFBGA100

01101: SMPS LQFP48/SMPS UQFN48
00001: SMPS WLCSP52
01000: SMPS LQFP64/SMPS UFBGA64
00011: SMPS WLCSP68-Q
00111: SMPS WLCSP68-G
01010: SMPS LQFP100/SMPS UFBGA100

Others: Reserved

55 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

56 Revision history

Table 584. Document revision history

Date	Revision	Changes
18-Feb-2025	1	Initial release.

Index

A

ADC_AWD1HTR 865
ADC_AWD1LTR 865
ADC_AWD2CR 864
ADC_AWD2HTR 866
ADC_AWD2LTR 866
ADC_AWD3CR 864
ADC_AWD3HTR 867
ADC_AWD3LTR 867
ADC_CALFACT 867
ADC_CFGR1 847
ADC_CFGR2 851
ADC_CR 844
ADC_DR 859
ADC_GCOMP 862
ADC_IER 842
ADC_ISR 839
ADC_JDRy 863
ADC_JSQR 859
ADC_OFCFGRy 861
ADC_OFRy 862
ADC_OR 869
ADC_PCSEL 855
ADC_SMPR1 853
ADC_SMPR2 854
ADC_SQR1 855
ADC_SQR2 856
ADC_SQR3 857
ADC_SQR4 858
ADCC_CCR 871
ADCC_CDR 873
ADCC_CDR2 873
ADCC_CSR 869
ADF_BSMX0CR 1008
ADF_CKGCR 1004
ADF_DFLT0CICR 1010
ADF_DFLT0CR 1008
ADF_DFLT0DR 1020
ADF_DFLT0IER 1013
ADF_DFLT0ISR 1014
ADF_DFLT0RSFR 1012
ADF_DLY0CR 1013
ADF_GCR 1003
ADF_SADANLVR 1019
ADF_SADCFGR 1017
ADF_SADCR 1015
ADF_SADSDLVR 1019
ADF_SITF0CR 1007
ADF_TRGISEL 1006
AES_CR 1113
AES_DINR 1116
AES_DOUTR 1117
AES_ICR 1124
AES_IER 1122
AES_ISR 1123
AES_IVR0 1119
AES_IVR1 1119
AES_IVR2 1119
AES_IVR3 1120
AES_KEYR0 1117
AES_KEYR1 1118
AES_KEYR2 1118
AES_KEYR3 1118
AES_KEYR4 1120
AES_KEYR5 1120
AES_KEYR6 1121
AES_KEYR7 1121
AES_SR 1115
AES_SUSPxR 1121
AP_BASE 2366
AP_BDn 2365
AP_CFG 2365
AP_CSW 2363
AP_DRWR 2365
AP_IDR 2366
AP_TAR 2364

B

BPU_CIDR0 2405
BPU_CIDR1 2405
BPU_CIDR2 2406
BPU_CIDR3 2406
BPU_COMPx 2402
BPU_CTRL 2401
BPU_DEVARCH 2402
BPU_DEVTYPE 2403
BPU_PIDR0 2403
BPU_PIDR1 2404
BPU_PIDR2 2404
BPU_PIDR3 2405
BPU_PIDR4 2403

C

C1ROM_CIDR3 . . . . . 2372, 2376
CCB_CR 1060

CCB_REFTAGRx 1063
CCB_SR 1061
COMP1_CSR 930
COMP2_CSR 932
CPUROM_CIDR0 2375
CPUROM_CIDR1 2375
CPUROM_CIDR2 2376
CPUROM_CIDR3 2376
CPUROM_MEMTYPER 2373
CPUROM_PIDR0 2373
CPUROM_PIDR1 2374
CPUROM_PIDR2 2374
CPUROM_PIDR3 2375
CPUROM_PIDR4 2373
CRC_CR 607
CRC_DR 606
CRC_IDR 606
CRC_INIT 608
CRC_POL 608
CRS_CFGR 449
CRS_CR 448
CRS_ICR 452
CRS_ISR 450
CTI_APPCLEA 2446
CTI_APPPULSE 2447
CTI_APPSET 2446
CTI_CHINSTATUS 2449
CTI_CHOUTSTATUS 2449
CTI_CIDR0 2453
CTI_CIDR1 2454
CTI_CIDR2 2454
CTI_CIDR3 2454
CTI_CONTROL 2445
CTI_DEVID 2450
CTI_DEVTYPE 2451
CTI_GATE 2450
CTI_INENx 2447
CTI_INTACK 2445
CTI_OUTENx 2448
CTI_PIDR0 2452
CTI_PIDR1 2452
CTI_PIDR2 2452
CTI_PIDR3 2453
CTI_PIDR4 2451
CTI_TRIGINSTATUS 2448
CTI_TRIGOUTSTATUS 2449

D

DAC_AUTOCR 916
DAC_CCR 911
DAC_CR 900
DAC_DHR12L1 905
DAC_DHR12L2 906
DAC_DHR12LD 908
DAC_DHR12R1 904
DAC_DHR12R2 906
DAC_DHR12RD 907
DAC_DHR8R1 905
DAC_DHR8R2 907
DAC_DHR8RD 908
DAC_DOR1 909
DAC_DOR2 909
DAC_MCR 912
DAC_SHHR 914
DAC_SHRR 915
DAC_SHSR1 913
DAC_SHSR2 914
DAC_SR 910
DAC_SWTRGR 903
DBGMCU_AHB1FZR 2461
DBGMCU_APB1HFZR 2459
DBGMCU_APB1LFZR 2458
DBGMCU_APB2FZR 2460
DBGMCU_APB3FZR 2460
DBGMCU_CIDR0 2465
DBGMCU_CIDR1 2465
DBGMCU_CIDR2 2465
DBGMCU_CIDR3 2466
DBGMCU_CR 2457
DBGMCU_DBG_AUTH_DEVICE 2462
DBGMCU_DBG_AUTH_HOST 2462
DBGMCU_IDCODE 2457
DBGMCU_PIDR0 2463
DBGMCU_PIDR1 2463
DBGMCU_PIDR2 2464
DBGMCU_PIDR3 2464
DBGMCU_PIDR4 2463
DBGMCU_SR 2461
DLYB_CFGR 753
DLYB_CR 752
DP_ABORT 2356
DP_CTRL_STAT 2357
DP_DLCR 2358
DP_DLPIDR 2359
DP_DPIDR 2356
DP_EVENTSTAT 2360
DP_RDBUFF 2361
DP_RESEND 2360
DP_SELECT 2360
DP_TARGETID 2359
DWT_CIDR0 2389
DWT_CIDR1 2389
DWT_CIDR2 2390

DWT_CIDR3 2390
DWT_COMPx 2382
DWT_CPICNT 2380
DWT_CTRL 2378
DWT_CYCCNT 2379
DWT_DEVARCH 2386
DWT_DEVTYPE 2387
DWT_EXCCNT 2380
DWT_FOLDCNT 2381
DWT_FUNCTION0 2382
DWT_FUNCTION1 2383
DWT_FUNCTION2 2384
DWT_FUNCTION3 2385
DWT_LSUCNT 2381
DWT_PCSR 2381
DWT_PIDR0 2387
DWT_PIDR1 2388
DWT_PIDR2 2388
DWT_PIDR3 2389
DWT_PIDR4 2387
DWT_SLPCNT 2380

E

ETM_AUTHSTAT 2425
ETM_CCCTLR 2412
ETM_CIDR0 2429
ETM_CIDR1 2429
ETM_CIDR2 2429
ETM_CIDR3 2430
ETM_CLAIMCLR 2425
ETM_CLAIMSET 2424
ETM_CNTRLDVR0 2413
ETM_CONFIGR 2409
ETM_DEVARCH 2426
ETM_DEVTYPE 2426
ETM_EVENTCTL0R 2409
ETM_EVENTCTL1R 2410
ETM_IDR0 2416
ETM_IDR1 2417
ETM_IDR10 2414
ETM_IDR11 2415
ETM_IDR12 2415
ETM_IDR13 2415
ETM_IDR2 2418
ETM_IDR3 2418
ETM_IDR4 2419
ETM_IDR5 2420
ETM_IDR8 2414
ETM_IDR9 2414
ETM_IMSPECR0 2416
ETM_PDCR 2423
ETM_PDSR 2424
ETM_PIDR0 2427
ETM_PIDR1 2427
ETM_PIDR2 2428
ETM_PIDR3 2428
ETM_PIDR4 2427
ETM_PRGCTLR 2408
ETM_RSCTLR2 2421
ETM_RSCTLR3 2421
ETM_SSCCR0 2422
ETM_SSCSR0 2422
ETM_SSPCICR0 2423
ETM_STALLCTLR 2411
ETM_STATR 2408
ETM_SYNCPR 2411
ETM_TRACEIDR 2412
ETM_VICTLR 2413
EXTI_EMR1 601
EXTI_EXTICRx 597
EXTI_FPR1 596
EXTI_FTSR1 594
EXTI_IMR1 600
EXTI_LOCKR 600
EXTI_PRIVCFGR1 597
EXTI_RPR1 595
EXTI_RTSR1 593
EXTI_SECCFGR1 596
EXTI_SWIER1 594

F

FDCAN_CCCR 2269
FDCAN_CKDIV 2294
FDCAN_CREL 2266
FDCAN_DBTP 2266
FDCAN_ECR 2274
FDCAN_ENDN 2266
FDCAN_HPMS 2285
FDCAN_IE 2280
FDCAN_ILE 2283
FDCAN_ILS 2282
FDCAN_IR 2277
FDCAN_NBTP 2270
FDCAN_PSR 2275
FDCAN_RWD 2268
FDCAN_RXF0A 2287
FDCAN_RXF0S 2286
FDCAN_RXF1A 2288
FDCAN_RXF1S 2287
FDCAN_RXGFC 2283
FDCAN_TDCR 2277
FDCAN_TEST 2267

FDCAN_TOCC 2273
FDCAN_TOCV 2274
FDCAN_TSCC 2272
FDCAN_TSCV 2272
FDCAN_TXBAR 2290
FDCAN_TXBC 2288
FDCAN_TXBCF 2292
FDCAN_TXBCIE 2293
FDCAN_TXBCR 2291
FDCAN_TXBRP 2289
FDCAN_TXBTIE 2292
FDCAN_TXBTO 2291
FDCAN_TXEFA 2294
FDCAN_TXEFS 2293
FDCAN_TXFQS 2289
FDCAN_XIDAM 2285
FLASH_ACR 238
FLASH_BOOT0R 253
FLASH_BOOT1R 254
FLASH_CR 245
FLASH_ECCCORR 248
FLASH_ECCDETR 249
FLASH_KEYR 239
FLASH_OEM1KEYR1 268
FLASH_OEM1KEYR2 269
FLASH_OEM1KEYR3 269
FLASH_OEM1KEYR4 270
FLASH_OEM2KEYR1 270
FLASH_OEM2KEYR2 271
FLASH_OEM2KEYR3 271
FLASH_OEM2KEYR4 272
FLASH_OEMKEYSR 272
FLASH_OPSR 250
FLASH_OPTKEYR 240
FLASH_OPTR 251
FLASH_PDKEY1R 241
FLASH_PDKEY2R 241
FLASH_PRIVBB1Rx 267
FLASH_PRIVBB2Rx 268
FLASH_PRIVCFGR 266
FLASH_SBOOT0R 255
FLASH_SCR 247
FLASH_SECBB1Rx 264
FLASH_SECBB2Rx 264
FLASH_SECHDPCR 265
FLASH_SECHDPEXTR 266
FLASH_SECWM1R1 256
FLASH_SECWM1R2 257
FLASH_SECWM2R1 260
FLASH_SECWM2R2 261
FLASH_SKEYR 240
FLASH_SR 242
FLASH_SSR 243
FLASH_WRP1AR 258
FLASH_WRP1BR 259
FLASH_WRP2AR 262
FLASH_WRP2BR 263

G

GPDMA_CxBR1 567-568
GPDMA_CxBR2 574
GPDMA_CxCR 557
GPDMA_CxDAR 572
GPDMA_CxFCR 555
GPDMA_CxLBAR 554
GPDMA_CxLLR 575, 577
GPDMA_CxSAR 571
GPDMA_CxSR 556
GPDMA_CxTR1 559
GPDMA_CxTR2 563
GPDMA_CxTR3 573
GPDMA_MISR 553
GPDMA_PRIVCFGR 552
GPDMA_RCFGLOCKR 552
GPDMA_SECCFGR 551
GPDMA_SMISR 554
GPIOx_AFRH 471
GPIOx_AFRL 471
GPIOx_BRR 472
GPIOx_BSRR 469
GPIOx_HSLVR 473
GPIOx_IDR 468
GPIOx_LCKR 470
GPIOx_MODER 465
GPIOx_ODR 468
GPIOx_OSPEEDR 466
GPIOx_OTYPER 466
GPIOx_PUPDR 467
GPIOx_SECCFGR 474
GTZC1_MPCBB1_CFGLOCKR1 182
GTZC1_MPCBB1_CR 181
GTZC1_MPCBB1_PRIVCFGRx 183
GTZC1_MPCBB1_SECCFGRx 182
GTZC1_MPCBB2_CFGLOCKR1 185
GTZC1_MPCBB2_CR 184
GTZC1_MPCBB2_PRIVCFGRx 186
GTZC1_MPCBB2_SECCFGRx 185
GTZC1_TZIC_FCR1 173
GTZC1_TZIC_FCR2 175
GTZC1_TZIC_FCR3 176
GTZC1_TZIC_FCR4 178
GTZC1_TZIC_IER1 160
GTZC1_TZIC_IER2 162

GTZC1_TZIC_IER3 163
GTZC1_TZIC_IER4 165
GTZC1_TZIC_SR1 166
GTZC1_TZIC_SR2 168
GTZC1_TZIC_SR3 170
GTZC1_TZIC_SR4 171
GTZC1_TZSC_CR 147
GTZC1_TZSC_PRIVCFGR1 154
GTZC1_TZSC_PRIVCFGR2 156
GTZC1_TZSC_PRIVCFGR3 157
GTZC1_TZSC_SECCFGR1 148
GTZC1_TZSC_SECCFGR2 150
GTZC1_TZSC_SECCFGR3 152

H

HASH_CR 1194
HASH_CSRx 1202
HASH_DIN 1196
HASH_HRAx 1199
HASH_HRx 1199
HASH_IMR 1200
HASH_SR 1200
HASH_STR 1197

I

I2C_AUTOCR 1856
I2C_CR1 1843
I2C_CR2 1846
I2C_ICR 1854
I2C_ISR 1851
I2C_OAR1 1848
I2C_OAR2 1848
I2C_PECR 1855
I2C_RXDR 1855
I2C_TIMEOUTR 1850
I2C_TIMINGR 1849
I2C_TXDR 1856
I3C_BCR 1963
I3C_CEVR 1950
I3C_CFGR 1928
I3C_CR 1924, 1926
I3C_CRCAPR 1966
I3C_DCR 1964
I3C_DEVR0 1953
I3C_DEVRx 1955
I3C_EPIDR 1969
I3C_EVR 1944
I3C_GETCAPR 1965
I3C_GETMXDSR 1967
I3C_IBIDR 1938
I3C_IER 1948
I3C_MAXRLR 1957
I3C_MAXWLR 1958
I3C_MISR 1952
I3C_RDR 1933
I3C_RDWR 1934
I3C_RMR 1943
I3C_SER 1941
I3C_SR 1940
I3C_TDR 1935
I3C_TDWR 1936
I3C_TGTTDR 1939
I3C_TIMINGR0 1959
I3C_TIMINGR1 1960
I3C_TIMINGR2 1962
ICACHE_CR 288
ICACHE_CRRx 291
ICACHE_FCR 290
ICACHE_HMONR 291
ICACHE_IER 290
ICACHE_MMONR 291
ICACHE_SR 289
ITM_CIDR0 2398
ITM_CIDR1 2399
ITM_CIDR2 2399
ITM_CIDR3 2399
ITM_DEVARCH 2395
ITM_DEVTYPE 2396
ITM_PIDR0 2397
ITM_PIDR1 2397
ITM_PIDR2 2397
ITM_PIDR3 2398
ITM_PIDR4 2396
ITM_STIMx 2393
ITM_TCR 2394
ITM_TER 2393
ITM_TPR 2394
IWDG_EWCR 1696
IWDG_KR 1692
IWDG_PR 1693
IWDG_RLR 1694
IWDG_SR 1694
IWDG_WINR 1695

L

LPTIM_ARR 1674
LPTIM_CCMR1 1676
LPTIM_CCR1 1673
LPTIM_CCR2 1679
LPTIM_CFGR 1669
LPTIM_CFGR2 1675
LPTIM_CNT 1674

LPTIM_CR 1672
LPTIM_RCR 1676
LPTIM4_DIER 1664
LPTIM4_ICR 1661
LPTIM4_ISR 1656
LPTIMx_DIER 1666-1667
LPTIMx_ICR 1662-1663
LPTIMx_ISR 1657, 1659
LPUART_AUTOCR 2115
LPUART_BRR 2103
LPUART_CR1 . . . . . 2091, 2094
LPUART_CR2 2097
LPUART_CR3 . . . . . 2099, 2101
LPUART_ICR 2112
LPUART_ISR 2104, 2109
LPUART_PRESC 2114
LPUART_RDR 2113
LPUART_RQR 2103
LPUART_TDR 2113

M

MCUROM_CIDR0 2370
MCUROM_CIDR1 2371
MCUROM_CIDR2 2371
MCUROM_CIDR3 2371
MCUROM_MEMTYPER 2368
MCUROM_PIDR0 2369
MCUROM_PIDR1 2369
MCUROM_PIDR2 2369
MCUROM_PIDR3 2370
MCUROM_PIDR4 2368

O

OCTOSPI_ABR 655
OCTOSPI_AR 649
OCTOSPI_CCR 651
OCTOSPI_CR 641
OCTOSPI_DCR1 643
OCTOSPI_DCR2 645
OCTOSPI_DCR3 646
OCTOSPI_DCR4 646
OCTOSPI_DLR 648
OCTOSPI_DR 649
OCTOSPI_FCR 648
OCTOSPI_HLCR 663
OCTOSPI_IR 654
OCTOSPI_LPTR 655
OCTOSPI_PIR 651
OCTOSPI_PSMAR 651
OCTOSPI_PSMKR 650
OCTOSPI_SR 647
OCTOSPI_TCR 654
OCTOSPI_WABR 662
OCTOSPI_WCCR 659
OCTOSPI_WIR 662
OCTOSPI_WPABR 659
OCTOSPI_WPCCR 656
OCTOSPI_WPIR 658
OCTOSPI_WPTCR 658
OCTOSPI_WTCR 661
OPAMP1_CSR 941
OPAMP1_LPOTR 943
OPAMP1_OTR 943
OPAMP2_CRS 944
OPAMP2_OTR 945

P

PKA_CLRFR 1239
PKA_CR 1235
PKA_SR 1237
PWR_APCR 345
PWR_BDCR 340
PWR_CR1 331
PWR_CR2 333
PWR_CR3 334
PWR_DBPR 340
PWR_I3CPUCR1 352
PWR_I3CPUCR2 353
PWR_PDCRA 346
PWR_PDCRB 347
PWR_PDCRG 350
PWR_PDCRH 351
PWR_PDCRx 349
PWR_PRIVCFGR 342
PWR_PUCRA 345
PWR_PUCRG 349
PWR_PUCRH 351
PWR_PUCRx 348
PWR_SECCFGR 341
PWR_SR 342
PWR_SVMCR 336
PWR_SVMSR 343
PWR_VOSR 335
PWR_WUCR1 338
PWR_WUCR2 338
PWR_WUCR3 339
PWR_WUSCR 344
PWR_WUSR 344

R

RAMCFG_M2PARKEYR 194
RAMCFG_M2WPR1 193

RAMCFG_M2WPR2 194
RAMCFG_MxERKEYR 194
RCC_AHB1ENR1 403
RCC_AHB1ENR2 406
RCC_AHB1RSTR1 397
RCC_AHB1SLPENR1 411
RCC_AHB1SLPENR2 415
RCC_AHB1STPENR1 420
RCC_AHB2ENR1 405
RCC_AHB2ENR2 406
RCC_AHB2RSTR1 398
RCC_AHB2RSTR2 399
RCC_AHB2SLPENR1 412
RCC_AHB2SLPENR2 414
RCC_AHB2STPENR1 421
RCC_APB1ENR1 407
RCC_APB1ENR2 409
RCC_APB1RSTR1 400
RCC_APB1RSTR2 401
RCC_APB1SLPENR1 415
RCC_APB1SLPENR2 417
RCC_APB1STPENR1 422
RCC_APB1STPENR2 423
RCC_APB2ENR 409
RCC_APB2RSTR 402
RCC_APB2SLPENR 418
RCC_APB2STPENR 424
RCC_APB3ENR 410
RCC_APB3RSTR 403
RCC_APB3SLPENR 419
RCC_APB3STPENR 425
RCC_BDCR 431
RCC_CCIPR1 426
RCC_CCIPR2 428
RCC_CCIPR3 430
RCC_CFGR1 388
RCC_CFGR2 390
RCC_CFGR3 391
RCC_CFGR4 392
RCC_CICR 396
RCC_CIER 393
RCC_CIFR 394
RCC_CR 380
RCC_CRRCR 387
RCC_CSR 433
RCC_ICSCR1 383
RCC_ICSCR2 386
RCC_ICSCR3 387
RCC_PRIVCFGR 436
RCC_SECCFGR 435
RNG_CR 1076
RNG_DR 1079
RNG_HTCRx 1080
RNG_NSCR 1080
RNG_SR 1078
RTC_ALRABINR 1757
RTC_ALRBBINR 1757
RTC_ALRMAR 1745
RTC_ALRMASSR 1747
RTC_ALRMBR 1748
RTC_ALRMBSSR 1749
RTC_CALR 1741
RTC_CR 1734
RTC_DR 1728
RTC_ICSR 1730
RTC_MISR 1751
RTC_PRER 1732
RTC_PRIVCFGR 1738
RTC_SCR 1754
RTC_SECCFGR 1739
RTC_SHIFTR 1742
RTC_SMISR 1752
RTC_SR 1750
RTC_SSR 1729
RTC_TAMPTSCR 1755
RTC_TR 1727
RTC_TSDR 1744
RTC_TSIDR 1756
RTC_TSSSR 1745
RTC_TSTR 1743
RTC_WPR 1740
RTC_WUTR 1733

S

SAES_CR 1167
SAES_DINR 1171
SAES_DOUTR 1172
SAES_ICR 1179
SAES_IER 1177
SAES_ISR 1178
SAES_IVR0 1174
SAES_IVR1 1174
SAES_IVR2 1174
SAES_IVR3 1175
SAES_KEYR0 1172
SAES_KEYR1 1173
SAES_KEYR2 1173
SAES_KEYR3 1173
SAES_KEYR4 1175
SAES_KEYR5 1175
SAES_KEYR6 1176
SAES_KEYR7 1176
SAES_SR 1170

SAES_SUSPxR 1176
SAI_ACLRFR 2215
SAI_ACR1 2205
SAI_ACR2 2207
SAI_ADR 2216
SAI_AFRCR 2209
SAI_AIM 2211
SAI_ASLOTR 2210
SAI_ASR 2213
SAI_BCLRFR 2226
SAI_BCR1 2216
SAI_BCR2 2219
SAI_BDR 2227
SAI_BFRCR 2221
SAI_BIM 2223
SAI_BSLOTR 2222
SAI_BSR 2224
SAI_PDMCR 2228
SAI_PDMDLY 2229
SDMMC_ACKTIMER 742
SDMMC_ARGR 727
SDMMC_CLKCR 725
SDMMC_CMDR 727
SDMMC_DCNTR 733
SDMMC_DCTRL 732
SDMMC_DLENR 731
SDMMC_DTIMER 730
SDMMC_FIFORx 746
SDMMC_ICR 737
SDMMC_IDMABAR 745
SDMMC_IDMABASER 744
SDMMC_IDMABSIZER 743
SDMMC_IDMACTRLR 742
SDMMC_IDMALAR 744
SDMMC_MASKR 739
SDMMC_POWER 724
SDMMC_RESPCMDR 729
SDMMC_RESPxR 730
SDMMC_STAR 734
SPI_AUTOCR 2163
SPI_CFG1 2153
SPI_CFG2 2156
SPI_CR1 2151
SPI_CR2 2153
SPI_CRCPOLY 2164
SPI_IER 2158
SPI_IFCR 2162
SPI_RXCRC 2166
SPI_RXDR 2164
SPI_SR 2159
SPI_TXCRC 2165
SPI_TXDR 2163
SPI_UDRDR 2166
SYSCFG_CCCR 487
SYSCFG_CCCSR 485
SYSCFG_CCVR 486
SYSCFG_CFGR1 480
SYSCFG_CFGR2 484
SYSCFG_CNSLCKR 482
SYSCFG_CSLCKR 483
SYSCFG_FPUIMR 481
SYSCFG_RSSCMDR 488
SYSCFG_SECCFGR 479

T

TAMP_BKPxR 1789
TAMP_COUNT1R 1788
TAMP_CR1 1776
TAMP_CR2 1777
TAMP_CR3 1778
TAMP_FLTCR 1779
TAMP_IER 1783
TAMP_MISR 1785
TAMP_PRIVCFGR 1782
TAMP_RPCFGR 1788
TAMP_SCR 1787
TAMP_SECCFGR 1780
TAMP_SMISR 1786
TAMP_SR 1784
TIM1_AF1 1376
TIM1_AF2 1379
TIM1_ARR 1362
TIM1_BDTR 1366
TIM1_CCER 1357
TIM1_CCMR1 . . . . . 1348, 1350
TIM1_CCMR2 1353-1354
TIM1_CCMR3 1372
TIM1_CCR1 1363
TIM1_CCR2 1363
TIM1_CCR3 1364
TIM1_CCR4 1365
TIM1_CCR5 1370
TIM1_CCR6 1371
TIM1_CNT 1361
TIM1_CR1 1334
TIM1_CR2 1335
TIM1_DCR 1381
TIM1_DIER 1343
TIM1_DMAR 1383
TIM1_DTR2 1373
TIM1_ECR 1374
TIM1_EGR 1347
TIM1_PSC 1361

TIM1_RCR 1362
TIM1_SMCR 1339
TIM1_SR 1344
TIM1_TISEL 1375
TIM15_AF1 1597
TIM15_AF2 1599
TIM15_ARR 1590
TIM15_BDTR 1592
TIM15_CCER 1586
TIM15_CCMR1 1582-1583
TIM15_CCR1 1591
TIM15_CCR2 1592
TIM15_CNT 1589
TIM15_CR1 1572
TIM15_CR2 1573
TIM15_DCR 1600
TIM15_DIER 1578
TIM15_DMAR 1601
TIM15_DTR2 1595
TIM15_EGR 1581
TIM15_PSC 1589
TIM15_RCR 1590
TIM15_SMCR 1576
TIM15_SR 1579
TIM15_TISEL 1596
TIMx_AF1 . . . . . 1494, 1622
TIMx_AF2 . . . . . 1495, 1625
TIMx_ARR 1488, 1520, 1616
TIMx_BDTR 1618
TIMx_CCER 1485, 1612
TIMx_CCMR1 . . . . . 1477, 1479, 1609-1610
TIMx_CCMR2 1481-1482
TIMx_CCR1 . . . . . 1488, 1617
TIMx_CCR2 1489
TIMx_CCR3 1490
TIMx_CCR4 1491
TIMx_CNT 1487, 1519, 1615
TIMx_CR1 . . . . . 1466, 1515, 1604
TIMx_CR2 . . . . . 1467, 1517, 1605
TIMx_DCR 1496, 1626
TIMx_DIER 1473, 1517, 1606
TIMx_DMAR 1497, 1627
TIMx_DTR2 1621
TIMx_ECR 1492
TIMx_EGR 1476, 1519, 1608
TIMx_OR1 1625
TIMx_PSC 1487, 1520, 1615
TIMx_RCR 1616
TIMx_SMCR 1469
TIMx_SR 1474, 1519, 1607
TIMx_TISEL 1493, 1622
TPIU_ACPR 2435
TPIU_CIDR0 2442
TPIU_CIDR1 2442
TPIU_CIDR2 2442
TPIU_CIDR3 2443
TPIU_CLAIMCLR 2438
TPIU_CLAIMSET 2437
TPIU_CSPSR 2434
TPIU_DEVID 2438
TPIU_DEVTYPE 2439
TPIU_FFCR 2436
TPIU_FFSR 2436
TPIU_PIDR0 2440
TPIU_PIDR1 2440
TPIU_PIDR2 2441
TPIU_PIDR3 2441
TPIU_PIDR4 2439
TPIU_PSCR 2437
TPIU_SPPR 2435
TPIU_SSPSR 2434
TSC_CR 1031
TSC_ICR 1035
TSC_IER 1034
TSC_IOASCR 1036
TSC_IOCCR 1037
TSC_IOGCSR 1038
TSC_IOGxCR 1038
TSC_IOHCR 1036
TSC_IOSCR 1037
TSC_ISR 1035

U

USART_AUTOCR 2059
USART_BRR 2041
USART_CR1 . . . . . 2023, 2027
USART_CR2 2030
USART_CR3 . . . . . 2034, 2038
USART_GTPR 2042
USART_ICR 2056
USART_ISR 2045, 2051
USART_PRESC 2058
USART_RDR 2057
USART_RQR 2044
USART_RTOR 2043
USART_TDR 2058
USB_BCDR 2330
USB_CHEP_RXTXBD_n 2344-2345
USB_CHEP_TXRXBD_n 2342
USB_CHEPnR 2331
USB_CNTR 2321
USB_DADDR 2328
USB_FNR 2328

USB_ISTR 2324
USB_LPMCSR 2329

V

VREFBUF_CCR 922
VREFBUF_CSR 921

W

WWDG_CFR 1703
WWDG_CR 1702
WWDG_SR 1704

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved