SR5 E1 Line - Offline self-test

Introduction

This technical note describes how to run the different built-in self-test (BIST) in the SR5E1 device.

The user can run distinct kinds of BIST:

- MBIST: RAM memory BIST
- CBIST: to detect errors in the comparator modules.

On the SR5E1 device, the user must run these self-tests during the startup phase. In this mode, the execution of these BISTs is transparent for the application.

The latest generations of STMicroelectronics microcontrollers (SPC58 and SR6/SR5) offer high flexibility in the self-test configuration. The user can configure:

1. Which BIST starts (MBIST, or CBIST)
2. At which frequency self-tests run
3. If BIST runs in parallel or sequential mode
4. A delay between the memory partitions if more than a single MBIST runs in parallel mode.

This document provides a complete example of offline self-tests, including CBIST and MBIST.

This technical note focus on the SR5E1 device, but most concepts are also valid for other devices of the SR5 family.

1 Overview

BISTs (built-in self-tests) detect latent faults in the device. The user must run it once per trip time.

The SRE1 device includes two different types of BIST:

- Memory BIST (MBIST): for volatile memories
- CBIST: for detecting faults, in the comparators.

BIST runs during reset phases through a set of DCFs (offline mode).

The self-test control unit (STCU3) hardware module controls the BIST execution.

Moreover, the STCU3 has two different watchdogs that monitor two reset phases. They are:

- Hardcoded watchdog monitors the “initialization” phase. The SSCM module dispatches the DCF value to the proper client, STCU3 included
- Watchdog timer (WDG): monitors the “self-test execution”

For this device, BIST runs only during the reset phase (that is, offline mode).

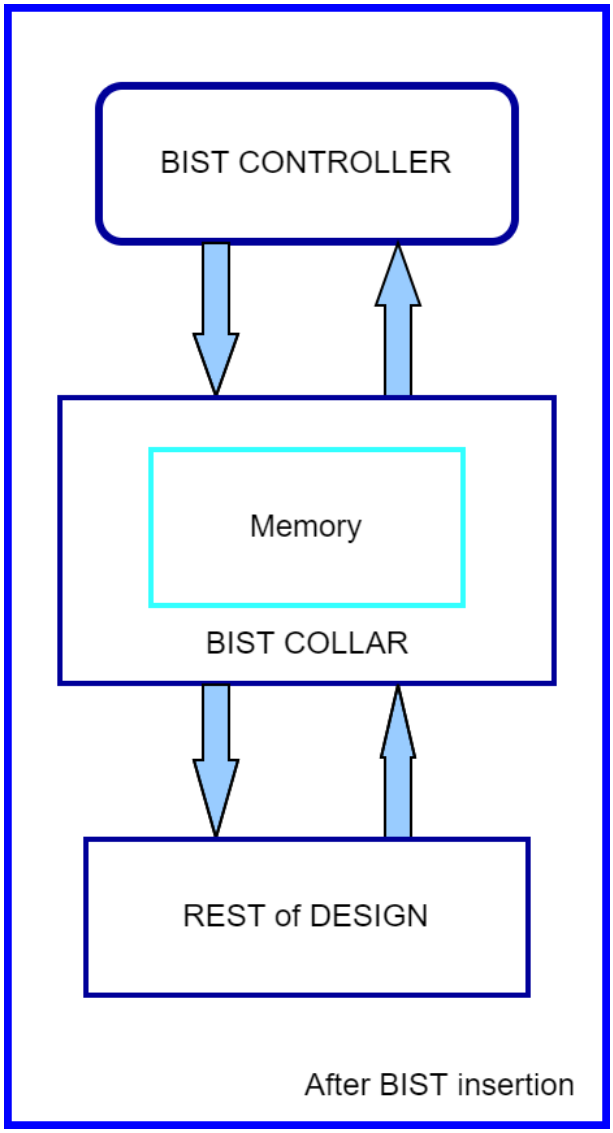
1.1 Memory BIST architecture (MBIST)

MBIST verifies the integrity of volatile memories.

Note: For non-volatile memories another type of test is available (that is, the array integrity check). The reference manual contains the details on this test.

The following figure shows the basic architecture of the MBIST.

Figure 1. **Basic MBIST architecture**



The BIST collar is a wrapper around the memory RAM that executes the instructions received from the controller to test the memory.

The SR5E1 devices have a single MBIST controller that includes only memory cuts of the volatile memories. Many details about the MBIST organization are reported in the device configuration chapter of the reference manual (see Appendix A.1 Reference documents).

This device has 58 memory cuts (from 0 to 57) inside the MBIST controller 0.

1.2 CBIST (Comparator BIST)

The CBIST controllers check the functionality of the comparators used by the lockstep architecture. The CBIST injects a fault in the comparator's input and verifies if the comparator detects it.

In this device, there are 10 CBIST (from 0 to 9).

Many details about the CBIST organization are reported in the device configuration chapter of the reference manual (see Appendix A.1 Reference documents).

2 BIST configurations

This paragraph describes which modules and clocks are involved in executing the BIST in offline mode.

2.1 Offline mode configuration

The SSCM configures the STCU3 during the boot phase, according to the related DCF records stored in the UTest sector of the NVM.

The main parameters to configure the offline self-test are:

- CBIST and MBIST scheduling activity
- Fault management
- Usage of the STCU3 CRC
- Clock management (clock of the peripheral and cluster domains)

Users must apply such configurations by storing the related DCF records in the UTest.

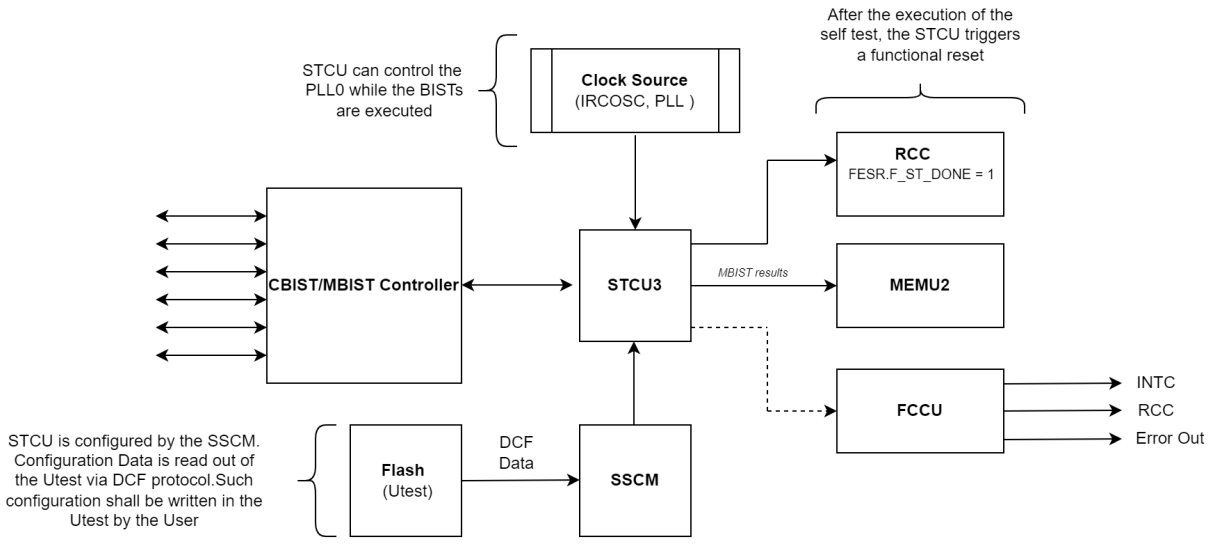
2.2 Modules used in the self-test phase

There are several hardware modules involved in the self-test execution in offline mode, for example:

- Reset and clock control module (RCC),
- C/MBIST controllers,
- System status and configuration module (SSCM)
- Self-test control unit (STCU3)
- Fault collection and control unit (FCCU)
- UTEST of NVM
- MEMU2 for the reporting of the MBIST faults

The following figure shows the interconnection among these modules.

Figure 2. Modules interconnection in offline mode



In offline mode, the BISTs run without any software intervention. During the reset phases, the SSCM loads the STCU3 configurations and triggers the start of the test.

The PLL must run at the maximum allowed frequency for the best timing performance.

The STCU3 forwards any BIST failure event to the FCCU. After BISTs execution, the STCU3 triggers a functional reset.

Note: In this case, the hardware sets the "ST_DONE" flag in the RCC.FESR register of the RCC module.

If the STCU3 detects an error during the execution of the MBIST, it logs the error in the MEMU2 peripherals.

2.3 Offline clock setting

Depending on the selected mode, the CBIST/MBIST can run with different clock configurations.

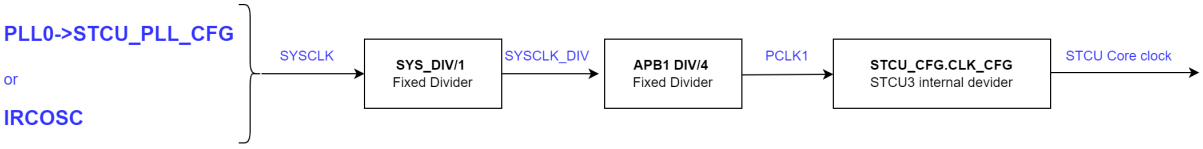
Two clock sources are available during the execution of the offline BIST:

1. IRCOSC
2. PLL0

Note that the XOSC clock source is not active during the execution of the offline BIST.

Besides the frequency limitation of each memory and logic part involved during the offline BIST, the clock for the STCU3 logic comes from the system clock (SYSCLK_DIV) that goes through two dividers configured by APB1_DIV and STCU_CFG.CLK_CFG (parameter of the **STCU_CFG** register).

Figure 3. STCU3 clock during the execution of the offline BIST



The PCLK1 clock drives the clock of the STCU3 and its internal watchdog.

Note: Check the datasheet (see Appendix A.1 Reference documents) to verify what is the max frequency of the device under use.

The clock source of the PCLK1 clock depends on the reset phases and STCU3 configuration.

The maximum frequency for the STCU3 Core clock is 50 MHz and the STCU_CFG.CLK_CFG.

Refer to the reference manual (see Appendix A.1 Reference documents) for the detailed clock tree and the maximum allowed clock frequencies.

2.4 PLLs configuration during offline self-test

BISTs can run with different clock configurations.

In offline mode, the STCU3 can run up to 50Mhz (STCU3 Core clock). The user can program this frequency by configuring the STCU_PLL0_CFG at the maximum frequency of 300 Mhz.

Note: Check the datasheet (see Appendix A.1 Reference documents) to verify what is the max frequency of the device under use.

A fixed divider (APB1_DIV) divides this frequency by 4.

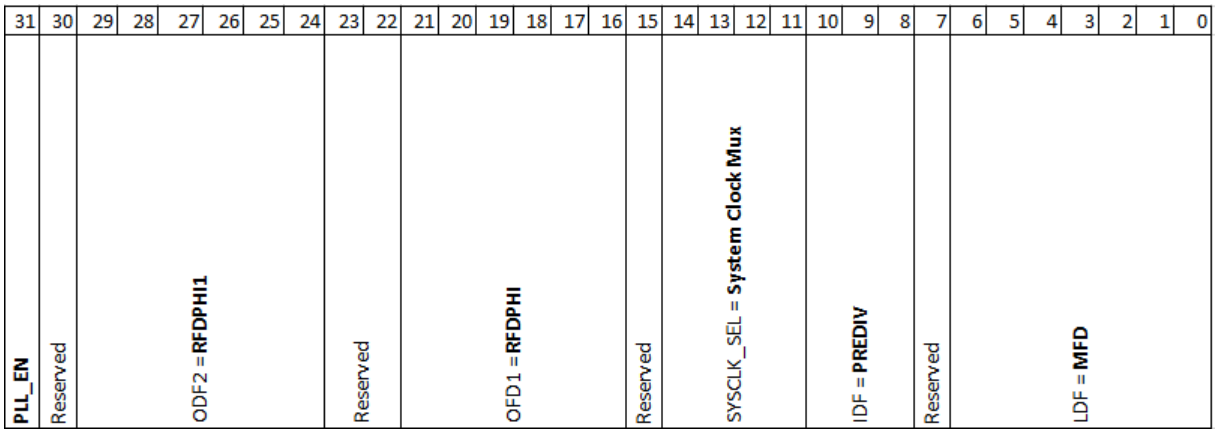
STCU3 can control the PLL0 during the off-line self-test operations depending on the value of the STCU_RUN[CBPLLEN] and STCU_RUN[MBPLLEN].

2.4.1 STCU_PLL0_CFG setup in offline mode

In offline mode, the output of the IRC oscillator must drive the PLL0.

The user can configure it by the STCU_PLL0_CFG register (see the following figure).

Figure 4. **STCU_PLL0_CGF register**



The clock reference for the PLL0 is the IRCOSC.

The user can select the PLL0 output by the SYSCLK_SEL field:

- *0 -> IRCOSC*
- *2 -> PLL0*

Based on the coefficient reported in the STCU_PLL0_CFG register, the PLL0 frequency is so calculated:

$$freq\_PLL0 = IRCOSC * MFD / (PREDIV * RFDPHI)$$

Where:

$$freq\_PLL0_{VCO} = IRCOSC * (2 * MFD) / PREDIV$$

The datasheet (see Appendix A.1 Reference documents) provides the maximum frequency of the freq_PLL0$_{VCO}$.

2.4.2 Recommended settings

The execution time of the MBIST depends on the clock frequency of each memory.

Note: It is also known as parallel mode.

Refer to the reference manual and data sheet (see Appendix A.1 Reference documents) to verify the maximum clock frequency of each memory.

The CBISTs can run at maximum speed.

2.5 BIST scheduling activity

STCU3 is flexible in terms of BIST scheduling. It allows programming the parallel t or sequential execution of the MBISTs or CBIST.

Note: It is also known as parallel mode.

Although the concurrent execution is shorter in terms of execution time, it causes higher power consumption. If this consumption is too high, the device cannot dissipate it.

Consequently, finding the right trade-off between execution time and power consumption is essential.

The user must indicate the first BIST that the STCU3 executes by the field.

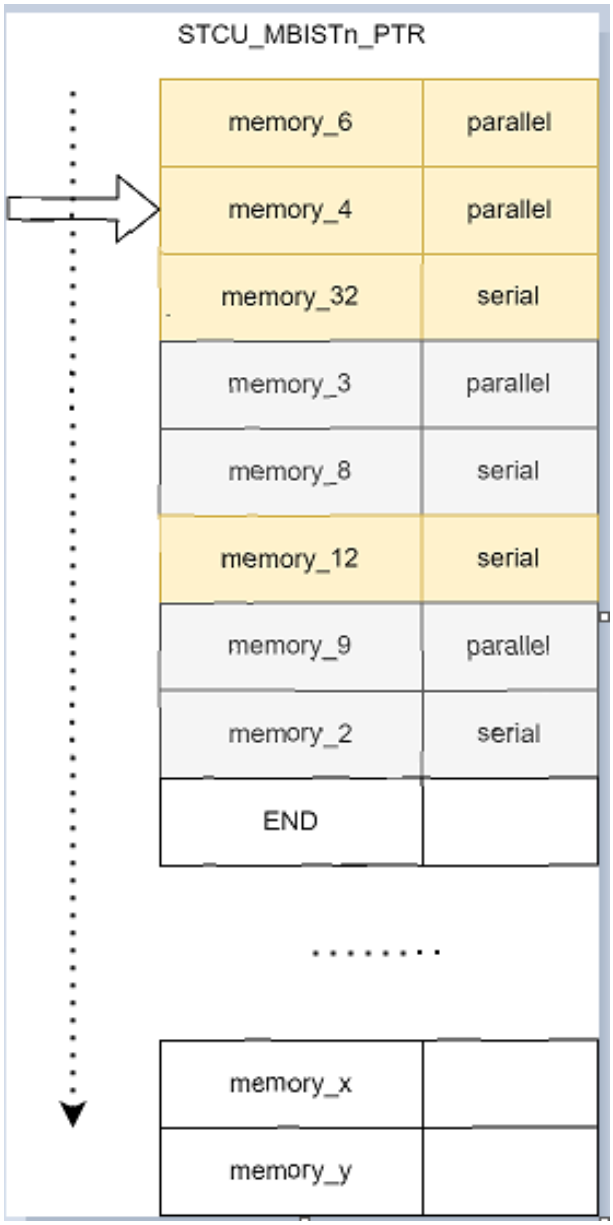
STCU_CFG[INIT_PTR]:

- 1 - > to start with MBIST
- 2 - > to start with CBIST.

Moreover, the user must configure the pointers to select the order of each test. There are three sets of registers that they must set:

- **STCU_MBISTn_PTR** for MBIST. It includes these specific flags
  - *BIST_TYPE* indicates the type of BIST for the subsequent BIST execution (M or CBIST). 0xF for the last BIST.
  - *BIST_CTRL_NUM* identifies the memory partition under test (see *MBIST partition* column in Table 1. List of memory cuts)
  - *BIST_PTR_VAL* indicates the memory under test (see Table 1. List of memory cuts)
  - *BIST_CFG:* indicates the mode–either sequential (0) or concurrent (1)–of the subsequent memory under test
- **STCU_CBISTn_PTR** for CBIST. It includes these specific flags
  - *BIST_TYPE*: indicates what is the type of BIST for the next BIST execution (CBIST). 0xF for the last BIST.
  - *BIST_CTRL_NUM* indicates the logical pointer to the current CBIST
  - *BIST_CFG:* indicates the mode (sequential (0) or concurrent (1))

Figure 5. Usage example of the MBISTn_PTR

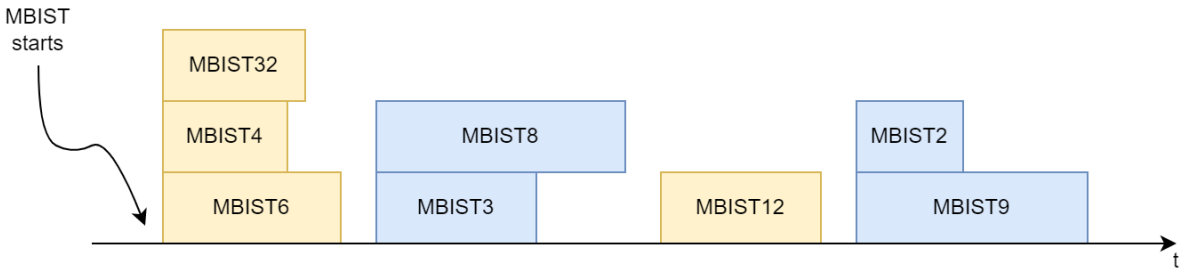


The figure above shows a simplified usage of the MBISTn_PTR to mix parallel and sequential execution modes. The STCU3 analyzes these pointers–and tests the related memories–starting from MBIST0_PTR until either:

- it arrives at the last pointer, or
- a pointer indicates the final MBIST in the *BIST_TYPE* field.

The figure below shows the execution of the MBIST configured in Figure 5. Usage example of the MBISTn_PTR.

Figure 6. Example of mixed concurrent and sequential execution mode



2.6 Example: how to run MBIST

This section configures the STCU3 registers to run 8 MBIST partitions of the device related to Core 1 with all memories tested in parallel.

This partition includes the following memory cuts:

Table 1. List of memory cuts

Memory name	STCU3 index
ICache_1	0
ICache_0	1
ICache_Tag_1	2
ICache_Tag_0	3
DCache_0_3	4
DCache_0_2	5
DCache_0_1	6
DCache_0_0	7

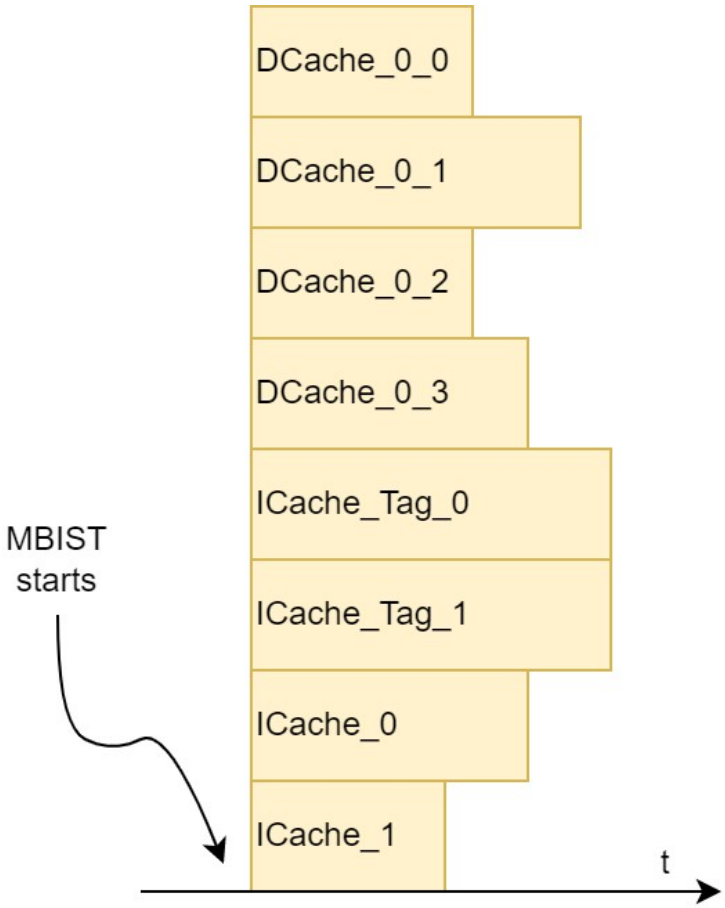
The following table shows the STCU3 configuration to run all MBIST memories in parallel mode. Figure 7. Test of MBIST in parallel mode shows how the STCU3 tests these memories.

Table 2. Example of MBIST settings

STCU registers	STCU setting
STCU_CFG = 0x 01000000	• Init_PTR = 1 -> MBIST
STCU_MBIST0_PTR = 0x01000001	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 0 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST1_PTR = 0x01000003	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 1 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST2_PTR = 0x01000005	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 2

STCU registers	STCU setting
	• BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST3_PTR = 0x01000007	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 3 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST4_PTR = 0x01000009	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 4 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST5_PTR = 0x0100000B	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 5 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST6_PTR = 0x0100000B	• BIST_TYPE = 1-> next BIST is another MBIST • BIST_CTRL_NUM = 0 -> number of MBIST controllers • BIST_PTR_VAL = 6 • BIST_CFG = 1 (concurrent/parallel mode)
STCU_MBIST7_PTR = 0x0F00000C	• BIST_TYPE = 0xF-> END flag value • BIST_CTRL_NUM = 0 -> number of MBIST controllerBIST_PTR_VAL = 7 • BIST_CFG = 0 (sequential mode)

Figure 7. **Test of MBIST in parallel mode**



3 STCU3 registers list

The hardware requires two keys to unlock the write access to the relevant STCU3 registers.
They are:

Table 3. STCU3 keys in offline mode

Mode	STCU_SCK value
Offline	Key 1 = 0xD3FEA98B Key 2 = 0x2C015674

The following paragraphs summarize the status of the STCU3 registers dedicated for offline test.
Each BIST (CBIST or MIST) has associated multiple status registers that report the result of the self-test execution.

3.1 MBIST status registers

There are 58 MBIST partitions (x: 0 to 57). Each partition has three associated status registers:

- STCU_MBx_STATUS indicates if MBISTx fails or not
- STCU_MBx_ENDFLAG indicates if MBIST x has been completed or not
- STCU_MBx_UFM configures the type of fault (recoverable or not recoverable) if the STCU detects a fault.

3.2 CBIST status register

There are 10 CBIST partitions (x: 0 to 9). Each partition has three associated status registers:

- STCU_CB_STATUS indicates if CBISTx fails or not
- STCU_CB_ENDFLAG indicates if CBISTx has been completed or not
- STCU_CB_UFM configures the type of fault (recoverable or not recoverable) if the STCU3 detects a fault.

4 BIST in offline mode

4.1 Reset and BIST execution

The sequence starts with a destructive reset triggered by an external or internal event.

The STCU3 executes the self-test during the *idle destructive* state of the reset sequence (see *reset sequence* figure in the reference manual see Appendix A.1 Reference documents).

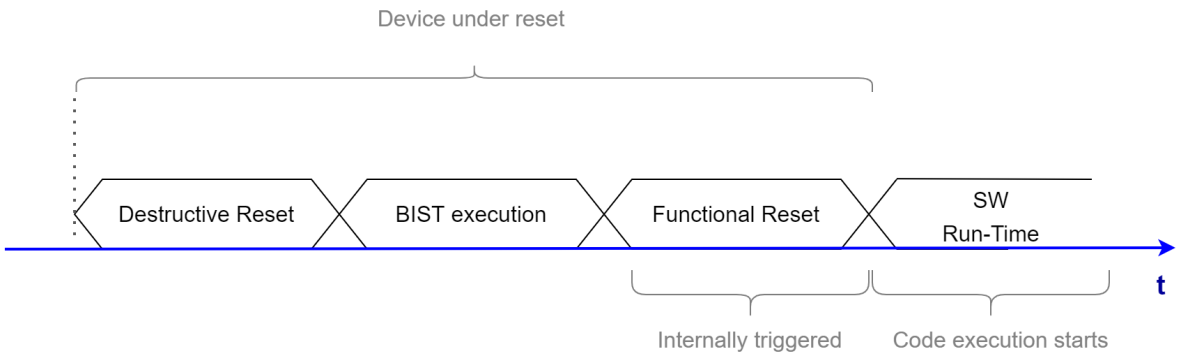
After an offline self-test, the hardware automatically triggers a functional reset (see the following figure).

Afterward, the device is ready to execute code in application mode.

During the BIST execution:

- a watchdog embedded in the STCU3 monitors the execution time of the self-test. The length of this phase is variable and depends on the self-test requirements. The user must estimate the watchdog timeout depending on the chosen configuration.
- The STCU3 executes all tests specified by the DCF records.
- The self-test engine updates the status registers and, if needed, triggers a fault.

Figure 8. Destructive and functional RESET sequence in case offline BISTs are enabled



4.2 STCU3 registers to setup in offline mode

After unlocking the STCU3, the user must configure some registers by DCF records.

Below is listed the essential configuration:

- *Watchdog timer (WDG)-> STCU_WDG*: to check if the self-test operations end within the assigned time slot.
  The watchdog timeout must be higher than the sum of the execution times of all selected BISTs.

To estimate the execution time, the user must consider that the offline self-test can be driven either by the IRCOSC or PLL0.

In case IRCOSC is used, the only parameter to configure is related to STCU_CFG[CLK_CFG]. These bits are the STCU3 core clock.

- Fault mapping of the STCU3 faults->STCU_ERR_FM. It defines if a fault is considered either unrecoverable or recoverable
- Index to schedule which BIST must start as the first one (MBIST or CBIST) >STCU_CFG[INIT_PTR].

- Select the test mode for MBIST by the STCU_MBn_TM register. ‘n’ indicates which MBIST controller runs (that is always 0 for such a device).

Table 4. MBIST test mode

Mode	Pmos	MBU bit
Autotest	0	1
	1	1
Reduced RunBIST	0	0
Full RunBIST	1	0

  - *Autotest:* it verifies the presence of multi bit errors in the RAM arrays
  - *Reduced RunBIST mode:* it includes the autotest and verifies the integrity of the RAM address logic
  - *Full RunBIST mode:* It includes the first two algorithms

- Fix the list of partitions to be executed by:
  - STCU_MBISTn_PTR
  - STCU_CBISTn_PTR

The methodology is to create an ordered list of pointers. Each pointer contains the BIST that the STCU3 executes (see Section 2.5 BIST scheduling activity).

- Start STCU->STCU_RUN[RUN]. This is the software trigger that starts the BIST execution.

4.3 DCF records to program in offline mode

DCF records are used to configure specific registers in the device during system boot while the reset signal is asserted.

A DCF record consists of 2 of 32bit contiguous words:

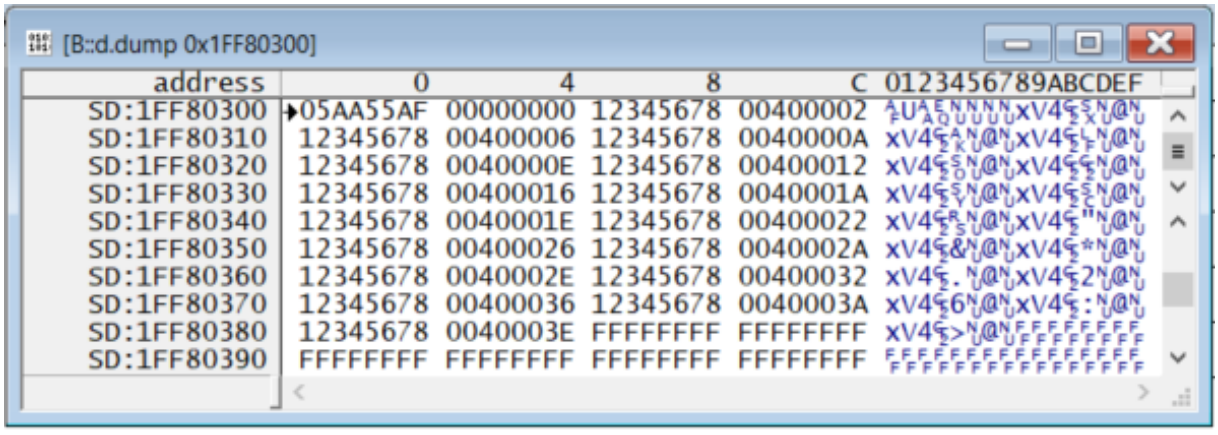
1. a pointer to the location of a register (this word includes also additional configuration), and
2. the data to be written to such a register.

The first 32 bits contain chip select, address, start, and stop bit information. The other 32 bits contain the data. All details about DCF and how it shall be used can be found in the section “Device configuration format (DCF) records” of the reference manual (see Appendix A.1 Reference documents).

In the SR5E1 device, UTEST memory is located between 0x1FF8_0000 to 0x1FF8_3FFF and the DCF location starts from 0x1FF8_0300.

The following figure shows the status of the Utest before storing any DCF records when the Life Cycle is STMicroelectronics production.

Figure 9. Utest before DCF programming



From 0x1FF8_0388 the user can start to program the DCF list for self-test or other device configurations setting.

4.4 Monitors during offline self-test

The self-test includes two phases:

1. Initialization (configuration loading): The SSCM configures the BISTs by programming the STCU3, and
2. Self-test execution: The STCU3 executes self-test.

Two different watchdogs monitor these phases.

• ***Hard-coded watchdog*** monitors the “initialization” phase.

It is a hardware watchdog configured at 0x3FF. The user cannot modify it.

The clock of the hard-coded watchdog depends on the RCOSC oscillator.

• ***Watchdog timer (WDG)*** monitors the “self-test execution.”

It is a hardware watchdog configurable by the user (STCU_WDG register). The user can check the status of the STCU3 WDG after the BIST execution in the STCU_ERR_STAT register (WDTO flag).

4.5 Hard-coded watchdog refresh during initialization

The hard-coded watchdog timeout is 0x3FF clock cycles.

The SSCM must periodically refresh the hard-coded watchdog by programming the STCU Key2.

To perform this operation, the user must interleave the list of DCF records with a write to the STCU Key2 register.

In the case of offline BIST, a single write of a DCF record takes around 17 clock cycles.

Since the hard-coded watchdog expires after 1024 clock cycles, the user must refresh it every 60 DCF records.

5 Offline BIST example

5.1 Configuration with CBIST and MBIST enabled

This paragraph shows a complete use case for the execution of CBIST and MBIST.

STCU_PLL0_CFG is set @300 Mhz (used for both, CBIST and MBIST).

According to the Figure 3. STCU3 clock during the execution of the offline BIST, this means that PCLK1 = 75 Mhz.

To avoid overcoming the frequency limitation of the STCU3 (50 Mhz), the STCU_CFG.CLK_CFG=1. This means that the STCU3 clock is PLCK1 divided by 2 (37.5 Mhz).

Table 5. List of DCF for CBIST/MBIST controller

Register	Value	DCF address	Comment
BIST configuration			
STCU_SKC	0xD3FEA98B	0x00080008	key 1
STCU_SKC	0x2C015674	0x00080008	key 2
STCU_CFG	0x02000001	0x00080010	Starting with CBIST - PCLK1 = 75 Mhz(SYS_CLK/4) - CFG_CLK =1- >STCU_CLOCK = 75 Mhz/2
STCU_MB0_TM	0x00000001	0x00080604	MBU =0- PMOS = 1-full Algo - 160N
STCU_PLL_CFG	0x8402124B	0x00080018	Sys_clock @ 300 Mhz
STCU_WDG	0x0000FFFF	0x0008003C	Setting watch at default value
STCU_ERR_FM	0x00000000	0x00080054	Configuring all other errors as recoverable
STCU_MB0_UFM0	0x00000000	0x00080844	Configuring all MBIST errors as recoverable
STCU_MB0_UFM1	0x00000000	0x00080848	Configuring all MBIST errors as recoverable
STCU_CB_UFM	0x00000000	0x00082F90	Configuring all CBIST errors as recoverable
CBIST list			
STCU_CBIST0_PTR	0x02000001	0x00084080	CBIST PTR 0 with next PTR in parallel mode
STCU_CBIST1_PTR	0x02010001	0x00084084	CBIST PTR 1 with next PTR in parallel mode
STCU_CBIST2_PTR	0x02020001	0x00084088	CBIST PTR 2 with next PTR in parallel mode
STCU_CBIST3_PTR	0x02030001	0x0008408C	CBIST PTR 3 with next PTR in parallel mode
STCU_CBIST4_PTR	0x02040001	0x00084090	CBIST PTR 4 with next PTR in parallel mode
STCU_CBIST5_PTR	0x02050001	0x00084094	CBIST PTR 5 with next PTR in parallel mode
STCU_CBIST6_PTR	0x02060001	0x00084098	CBIST PTR 6 with next PTR in parallel mode
STCU_CBIST7_PTR	0x02070001	0x0008409C	CBIST PTR 7 with next PTR in parallel mode
STCU_CBIST8_PTR	0x02080001	0x000840A0	CBIST PTR 8 with next PTR in parallel mode

Register	Value	DCF address	Comment
STCU_CBIST9_PTR	0x01090000	0x000840A4	CBIST PTR 9, with next PTR in sequential mode
MBIST list			
STCU_SKC	0x2C015674	0x00080008	key 2 - refresh
STCU_MBIST0_PTR	0x01000001	0x00083040	MBIST PTR 0, with next PTR in parallel mode
STCU_MBIST1_PTR	0x01000003	0x00083044	MBIST PTR 1, with next PTR in parallel mode
STCU_MBIST2_PTR	0x01000005	0x00083048	MBIST PTR 2, with next PTR in parallel mode
STCU_MBIST3_PTR	0x01000007	0x0008304C	MBIST PTR 3, with next PTR in parallel mode
STCU_MBIST4_PTR	0x01000009	0x00083050	MBIST PTR 4, with next PTR in parallel mode
STCU_MBIST5_PTR	0x0100000B	0x00083054	MBIST PTR 5, with next PTR in parallel mode
STCU_MBIST6_PTR	0x0100000D	0x00083058	MBIST PTR 6, with next PTR in parallel mode
STCU_MBIST7_PTR	0x0100000F	0x0008305C	MBIST PTR 7, with next PTR in parallel mode
STCU_MBIST8_PTR	0x01000011	0x00083060	MBIST PTR 8, with next PTR in parallel mode
STCU_MBIST9_PTR	0x01000013	0x00083064	MBIST PTR 9, with next PTR in parallel mode
STCU_MBIST10_PTR	0x01000015	0x00083068	MBIST PTR 10, with next PTR in parallel mode
STCU_MBIST11_PTR	0x01000017	0x0008306C	MBIST PTR 11, with next PTR in parallel mode
STCU_MBIST12_PTR	0x01000019	0x00083070	MBIST PTR 12, with next PTR in parallel mode
STCU_MBIST13_PTR	0x0100001B	0x00083074	MBIST PTR 13, with next PTR in parallel mode
STCU_MBIST14_PTR	0x0100001D	0x00083078	MBIST PTR 14, with next PTR in parallel mode
STCU_MBIST15_PTR	0x0100001F	0x0008307C	MBIST PTR 15, with next PTR in parallel mode
STCU_MBIST16_PTR	0x01000021	0x00083080	MBIST PTR 16, with next PTR in parallel mode
STCU_MBIST17_PTR	0x01000023	0x00083084	MBIST PTR 17, with next PTR in parallel mode
STCU_MBIST18_PTR	0x01000025	0x00083088	MBIST PTR 18, with next PTR in parallel mode
STCU_MBIST19_PTR	0x01000027	0x0008308C	MBIST PTR 19, with next PTR in parallel mode
STCU_MBIST20_PTR	0x01000029	0x00083090	MBIST PTR 20, with next PTR in parallel mode
STCU_MBIST21_PTR	0x0100002B	0x00083094	MBIST PTR 21, with next PTR in parallel mode
STCU_MBIST22_PTR	0x0100002D	0x00083098	MBIST PTR 22, with next PTR in parallel mode

Register	Value	DCF address	Comment
STCU_MBIST23_PTR	0x0100002F	0x0008309C	MBIST PTR 23, with next PTR in parallel mode
STCU_MBIST24_PTR	0x01000031	0x000830A0	MBIST PTR 24, with next PTR in parallel mode
STCU_MBIST25_PTR	0x01000033	0x000830A4	MBIST PTR 25, with next PTR in parallel mode
STCU_MBIST26_PTR	0x01000035	0x000830A8	MBIST PTR 26, with next PTR in parallel mode
STCU_MBIST27_PTR	0x01000037	0x000830AC	MBIST PTR 27, with next PTR in parallel mode
STCU_MBIST28_PTR	0x01000039	0x000830B0	MBIST PTR 28, with next PTR in parallel mode
STCU_MBIST29_PTR	0x0100003B	0x000830B4	MBIST PTR 29, with next PTR in parallel mode
STCU_MBIST30_PTR	0x0100003D	0x000830B8	MBIST PTR 30, with next PTR in parallel mode
STCU_MBIST31_PTR	0x0100003F	0x000830BC	MBIST PTR 31, with next PTR in parallel mode
STCU_MBIST32_PTR	0x01000041	0x000830C0	MBIST PTR 32, with next PTR in parallel mode
STCU_MBIST33_PTR	0x01000043	0x000830C4	MBIST PTR 33, with next PTR in parallel mode
STCU_MBIST34_PTR	0x01000045	0x000830C8	MBIST PTR 34, with next PTR in parallel mode
STCU_SKC	0x2C015674	0x00080008	key 2 - refresh
STCU_MBIST35_PTR	0x01000047	0x000830CC	MBIST PTR 35, with next PTR in parallel mode
STCU_MBIST36_PTR	0x01000049	0x000830D0	MBIST PTR 36, with next PTR in parallel mode
STCU_MBIST37_PTR	0x0100004B	0x000830D4	MBIST PTR 37, with next PTR in parallel mode
STCU_MBIST38_PTR	0x0100004D	0x000830D8	MBIST PTR 38, with next PTR in parallel mode
STCU_MBIST39_PTR	0x0100004F	0x000830DC	MBIST PTR 39, with next PTR in parallel mode
STCU_MBIST40_PTR	0x01000051	0x000830E0	MBIST PTR 40, with next PTR in parallel mode
STCU_MBIST41_PTR	0x01000053	0x000830E4	MBIST PTR 41, with next PTR in parallel mode
STCU_MBIST42_PTR	0x01000055	0x000830E8	MBIST PTR 42, with next PTR in parallel mode
STCU_MBIST43_PTR	0x01060057	0x000830EC	MBIST PTR 43, with next PTR in parallel mode
STCU_MBIST44_PTR	0x01000059	0x000830F0	MBIST PTR 44, with next PTR in parallel mode
STCU_MBIST45_PTR	0x0100005B	0x000830F4	MBIST PTR 45, with next PTR in parallel mode
STCU_MBIST46_PTR	0x0100005D	0x000830F8	MBIST PTR 46, with next PTR in parallel mode
STCU_MBIST47_PTR	0x0100005F	0x000830FC	MBIST PTR 47, with next PTR in parallel mode

Register	Value	DCF address	Comment
STCU_MBIST48_PTR	0x01000061	0x00083100	MBIST PTR 48, with next PTR in parallel mode
STCU_MBIST49_PTR	0x01060063	0x00083104	MBIST PTR 49, with next PTR in parallel mode
STCU_MBIST50_PTR	0x01000065	0x00083108	MBIST PTR 50, with next PTR in parallel mode
STCU_MBIST51_PTR	0x01000067	0x0008310C	MBIST PTR 51, with next PTR in parallel mode
STCU_MBIST52_PTR	0x01000069	0x00083110	MBIST PTR 52, with next PTR in parallel mode
STCU_MBIST53_PTR	0x0106006B	0x00083114	MBIST PTR 53, with next PTR in parallel mode
STCU_MBIST54_PTR	0x0100006D	0x00083118	MBIST PTR 54, with next PTR in parallel mode
STCU_MBIST55_PTR	0x0100006F	0x0008311C	MBIST PTR 55, with next PTR in parallel mode
STCU_MBIST56_PTR	0x01000071	0x00083120	MBIST PTR 56, with next PTR in parallel mode
STCU_MBIST57_PTR	0x01000072	0x00083124	Last PTR
Run BIST			
STCU_RUN	0x00000501	0x00080000	Start the test with IRCOSC for CBIST and PLL for MBIST

After a destructive reset of the device:

1. the SSCM loads the DCF client value in the STCU3
2. The STCU3 runs the configured test and provides the result.

The user must check that the CBIST status registers:

- STCU_CB_STATUS0 = 0x3FF
- STCU_CB_ENDFLAG0 = 0x3FF

Then the MBIST status registers:

- STCU_MB0_STATUS0 = 0xFFFF_FFFF
- STCU_MB0_ENDFLAG0 = 0xFFFF_FFFF
- STCU_MB0_STATUS1 = 0x03FF_FFFF
- STCU_MB0_ENDFLAG1 = 0x03FF_FFFF

And at least the global status error:

- STCU_ERR_STATUS = 0x0000_0000

5.2 How to change the DCF STCU3 configuration

This paragraph explains the correct procedure to update a STCU3 configuration if the user has already programmed the DCF record related to the STCU3 with a different configuration. This procedure is helpful during preliminary development stages when the user needs to identify the self-test configuration that fits with its application.

The flow to program STCU3 starts by providing the unlock keys and the rest of the configuration.

Unlock keys are fixed keys whose values are in the STCU3 chapter of the reference manual.

The STCU3 loads the configuration only if the user provides the correct unlock keys. If the unlock keys are incorrect, the STCU3 ignores the configuration that follows.

The user shall pass the key values to the STCU3 by writing them into the UTest sector of the Flash as DCF records

A problem occurs when the user needs to change the STCU3 configuration. It means the user has programmed at least a pair of unlock keys into UTEST.

STCU3 considers only the first valid unlock keys and the configuration that follows.

After the first configuration, the STCU3 does not accept other settings, even if the provided unlock keys are correct.

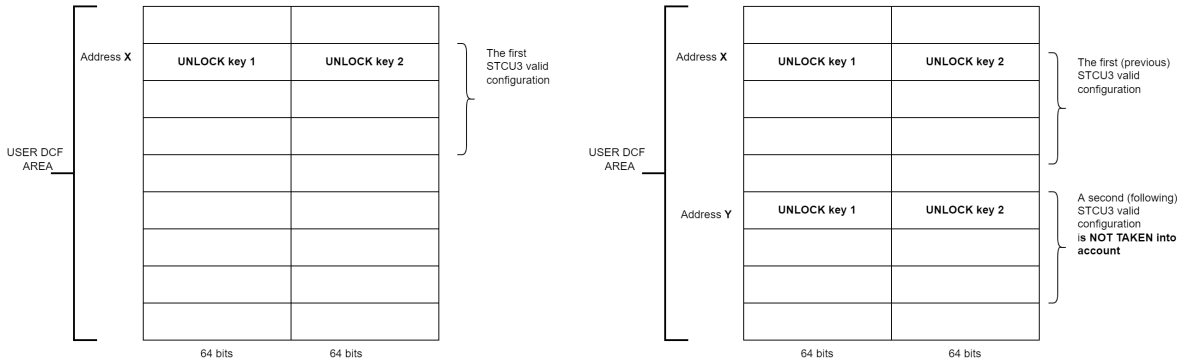
The following figure shows two different configurations of the STCU3. Both configurations start with the correct unlock keys:

1. The first configuration starts at Address X, and
2. The second configuration starts at Address Y.

Even if the second configuration starts with correct unlock keys, the STCU3 ignores it. As a result, the STCU3 runs the BIST according with the configuration that starts at Address X.

Next section explains how to instruct the STCU3 to load the second configuration and discard the first one.

Figure 10. STCU3 UNLOCK Keys organization



5.2.1 Using invalidation keys

As explained in the previous paragraph, simply writing a second STCU3 configuration at a different location does not cause any change in the configuration that the STCU3 loads (i.e., the first one).

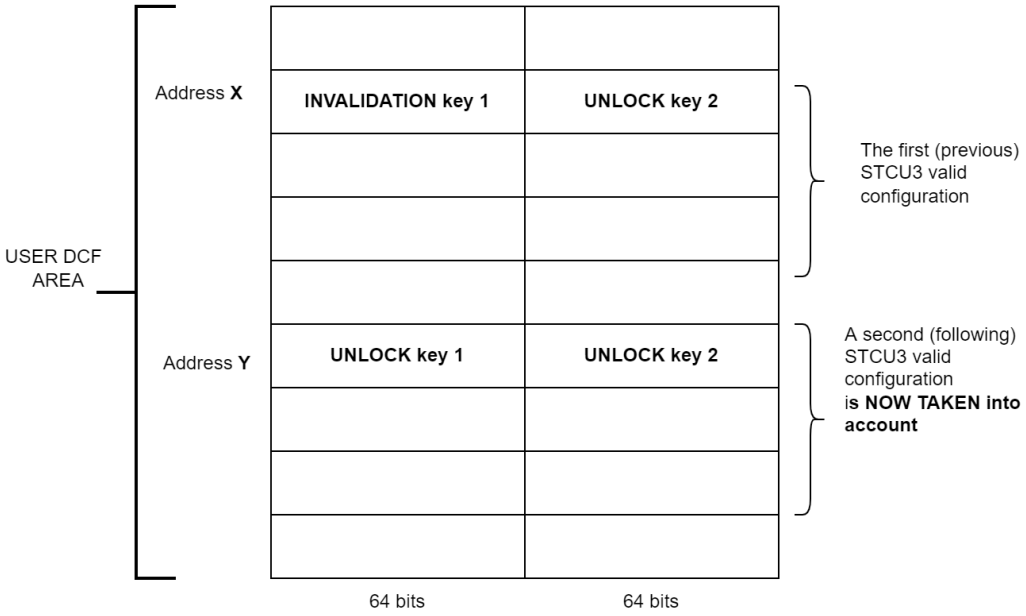
Since the UTEST sector is not erasable (OTP native), users must invalidate the first configuration if they want to change the STCU3 configuration. It means *invalidating* the first instance of the unlock keys.

Therefore, STCU3[RM1] reads unlock keys that are not correct and ignores the following configuration.

If the user *invalidates* one of the two unlock keys at address X, the STCU3 ignores the first configuration and loads the second one (see the following figure).

If needed, the user can repeat the process by adding a third configuration and *invalidating* the unlock key at address Y.

Figure 11. **Make unlock key in/valid**



Invalidating means overwriting the UNLOCK key with some incorrect keys. The user, however, must do this operation carefully due to the ECC.

The ECC protects the UTest content. The UTest contains not only the data, but also some redundant bits. The hardware can correct or detect some errors by comparing the data against the redundant bits. This process is transparent from the user's standpoint.

For this reason, if the user overwrites an UNLOCK key (STCU3 KEY 1 or STCU3 KEY 2) with random data, there is the risk that the ECC logic detects one or more not correctable errors. Under these circumstances, the sample doesn't start.

Consequently, the user must use specific values that don't cause ECC errors to invalidate the Unlock key.

STCU3 configuration can be invalidated by overwriting the STCU3 KEY1 with the proper value shown in the following table (STCU3 KEY 2 can remain the same).

Table 6. **STCU3 Key 1 configuration**

	Key1	Complete DCF Key1
UNLOCK	0xD3FEA98B	0x00080008D3FEA98B
INVALIDATION	0xD3BCA98B	0x000800080050a801

5.2.2 Limitation and prerequisites

- If the user invalidates the STCU3 Key1 and key2[RM1] , the FCCU signals an SSCM_XFER_FLASH_ERR_MEMORY_ERR [RM2] [FMGC3]
- The user shall use the INVALIDATION Keys only in case valid UNLOCK Keys are stored in UTEST and a new configuration must be applied
- INVALIDATION keys must be written at the exact positions of "UNLOCK Key (i.e., overwrite the UNLOCK key)"
- These INVALIDATION keys are not effective once the sample is programmed as "OPP" (Over Program Protection)
- STCU3 does not allow overwriting BYPASS. So key invalidation is required if we have already bypassed STCU3 and need to re-program it for a valid BISTs run.

6 Summary

This document describes how to perform the offline self-test on the SR5E1 device.

SR5E1 is composed by:

- 58 memory cuts
- 10 CBIST

Described concepts can be easily reused for other STMicroelectronics devices.

This document provides an example to speed up the implementation of the different available BIST.

Appendix A Further information

A.1 Reference documents

Table 7. **Reference documents**

Document name	Title
RM0483	SR5E1x 32-bit Arm® Cortex® -M7 architecture microcontroller for electrical vehicle applications
DS13808	SR5 E1 line of Stellar electrification MCUs — 32-bit Arm ® Cortex® -M7 automotive MCU 2x cores, 300 MHz, 2 MB Flash, rich analog, 104 ps 24 ch high-resolution timer, HSM, ASIL-D

A.2 Acronyms

Table 8. **Acronyms**

Acronym	Name
DCF	Device configuration format (DCF) records
UTest	User test flash block
STCU3	Self-test control unit
SSCM	System status and configuration module
RCC	Reset and clock control
FCCU	Fault collection and control unit
BIST	Generic term to indicate MBIST, LBIST or/and CBIST
MEMU2	Memory error management
IRCOSC	Internal RC oscillator

Revision history

Table 9. **Document revision history**

Date	Revision	Changes
12-Oct-2023	1	Initial release.

Contents

List of tables

List of figures