SR5 E1 line - multicore overview

Introduction

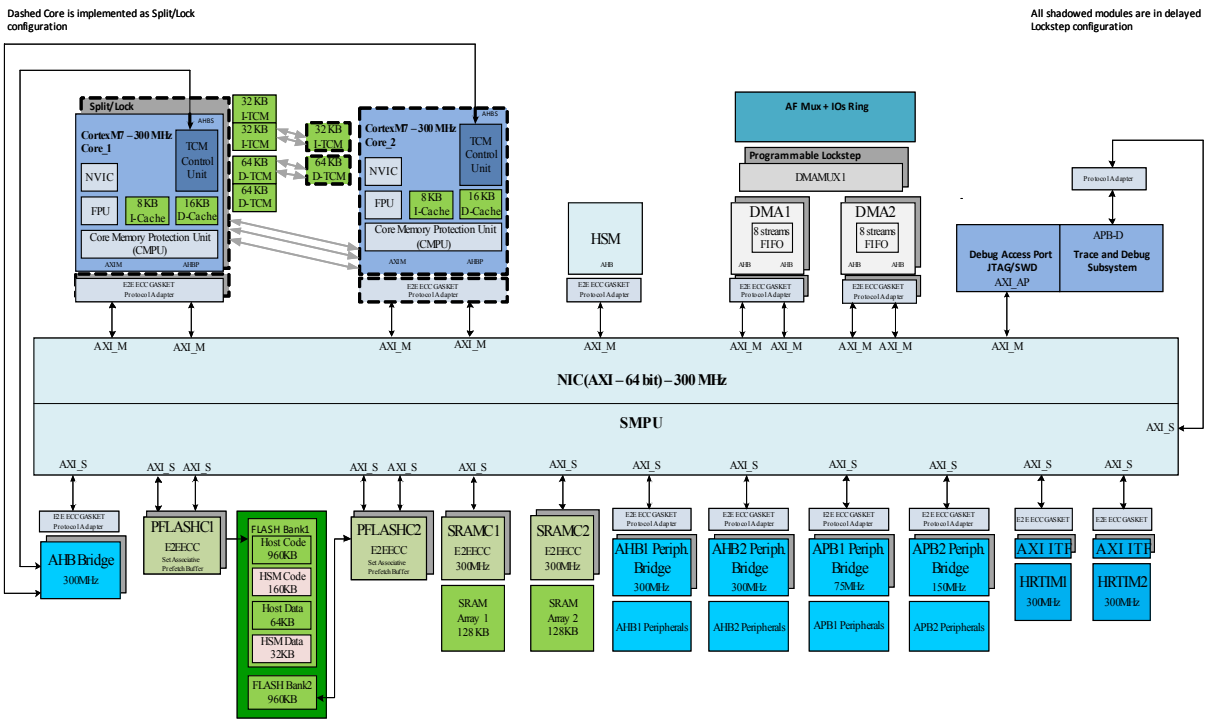
This document provides a brief overview of the multicore mechanism of the microcontrollers of the SR5 E1 line.

SR5 E1 line microcontrollers belong to a new generation of microcontroller family built on Arm® Cortex®-M7 cores, bringing a new level of performance, functional safety and security for real-time automotive applications. The architecture is split between one cluster domain that performs various computational and control functions and an HSM domain that is responsible for security cryptographic services. The cluster domain has two Arm® Cortex®-M7 cores that can work either in split or lockstep modes. Each core is identical in implementation and contains the same amount of caches and tightly coupled memories (TCM) that can be fully accessed even if the cores are working in lock-step modes. The HSM includes one Arm® Cortex®-M0+.

Table 1. SR5 E1 processor cores

Core	Type
Cortex® -M7 Cluster 0	
Main Core_1 (Boot)	Arm® Cortex® -M7
Main/Checker Core_2	Arm® Cortex® -M7
Hardware security module	
Arm® Cortex®-M0+	

Figure 1. SR5 E1 block diagram



The SR5E1 devices include:

- 256 Kbytes of system SRAM

- 128 Kbytes of data TCM RAM (DTCM)
- 64 Kbytes of instruction TCM RAM (ITCM)

The embedded system RAM is divided in two blocks:

- AXI SRAM1, 128 Kbytes is mapped at the address 0x2400_0000 and accessible by all system controllers.
- AXI SRAM2, 128 Kbytes is mapped at the address 0x2402_0000 and accessible by all system controllers.

Each Cortex®-M7 core has dedicated TCM SRAMs. The access is summarized in the table below.

Table 2. TCMs controller access

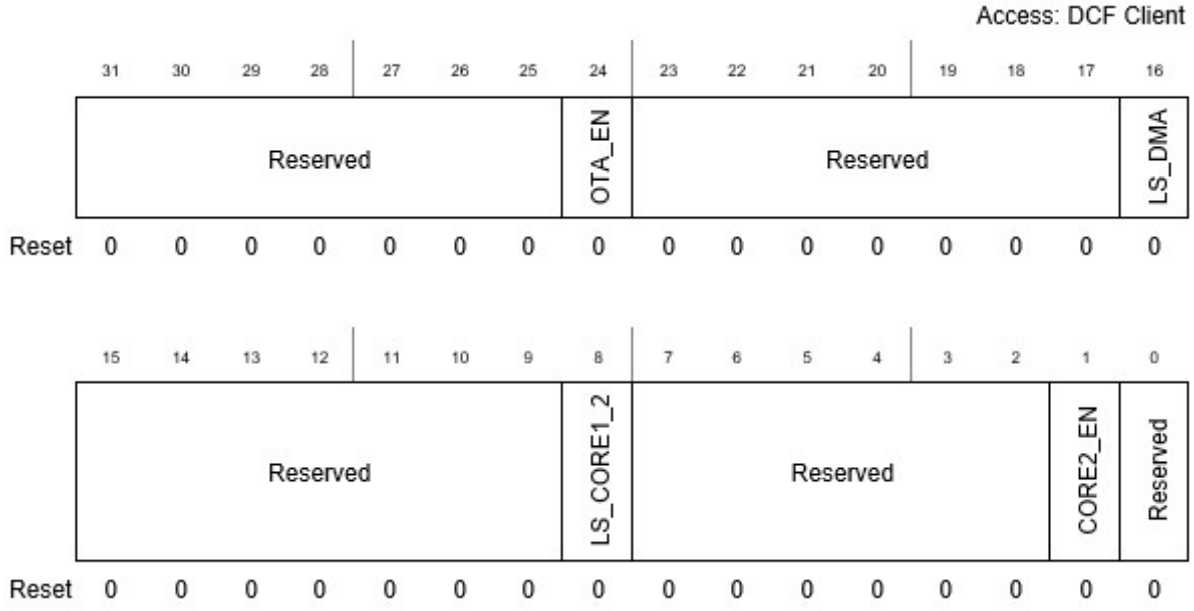
Memory	Address	Size	Access		
			Core 1	Core 2	DMA
ITCM1	0x5A000000	32 KB	Direct	AHBS	AHBS
ITCM2	0x5A040000	32 KB	AHBS	Direct	AHBS
DTCM1	0x5C000000	64 KB	Direct	AHBS	AHBS
DTCM2	0x5C040000	64 KB	AHBS	Direct	AHBS

When Core 1 and Core 2 are in lock-step mode, ITCM2 and DTCM2 are connected to Core 1 TCM interfaces and remapped offering a continuous range of 128 Kbytes of DTCM and 64 Kbytes of ITCM to Core 1.

1 Core modes

The bits CORE2_EN and LS_CORE1_2 in the global system configuration DCF register control the configuration of the two Arm® Cortex®-M7 cores.

Figure 2. SR5 E1 global system configuration DCF



By default (CORE2_EN = 0, LS_CORE1_2 = 0), the two Arm® Cortex®-M7 cores are configured in split mode and only Core 1 (main core) is enabled by hardware after the reset. This configuration requires the Core 1 user VTOR address (C1_VTOR) DCF is updated with the address of the Core 1 vector table for user application.

In split mode Core 2 can be woken-up by Core 1 or can be enabled by hardware via a dedicated DCF. The process of waking-up the Core 2 from Core 1 requires the settings of the reset and clock module (RCC) registers as described in the following steps:

- Update the Core 2 vector table offset init register (C2_VTOR_INIT_REG) with the address of the vector table of the Core 2.
- Release Core 2 reset setting to 1 the bit C2_RES_RELEASE in the Core 2 boot control register (C2_BOOT_CTRL_REG).
- Release Core 2 wait setting to 1 the bit C2_CPU_WAIT_RELEASE in the Core 2 boot control register (C2_BOOT_CTRL_REG).

Following an example of C code that enables the Core 2 from the Core 1:

```
RCC->C2_VTOR_INIT_REG  = 0x080F0000UL;
RCC->C2_BOOT_CTRL_REG |= (RCC_C2_BOOT_CTRL_REG_C2_RES_RELEASE);
RCC->C2_BOOT_CTRL_REG |= (RCC_C2_BOOT_CTRL_REG_C2_CPU_WAIT_RELEASE);
```

Moreover, the enabling by hardware of Core 2 in split mode is done by setting to 1 the bit CORE2_EN. This configuration also requires the Core 2 user VTOR address (C2_VTOR) DCF was updated with the address of the vector table of the Core 2.

Instead, the lockstep of the two cores is enabled setting to 1 the bit LS_CORE1_2. This configuration requires that only C1_VTOR DCF was updated with the address of the common vector table.

2 Core synchronization and triggering

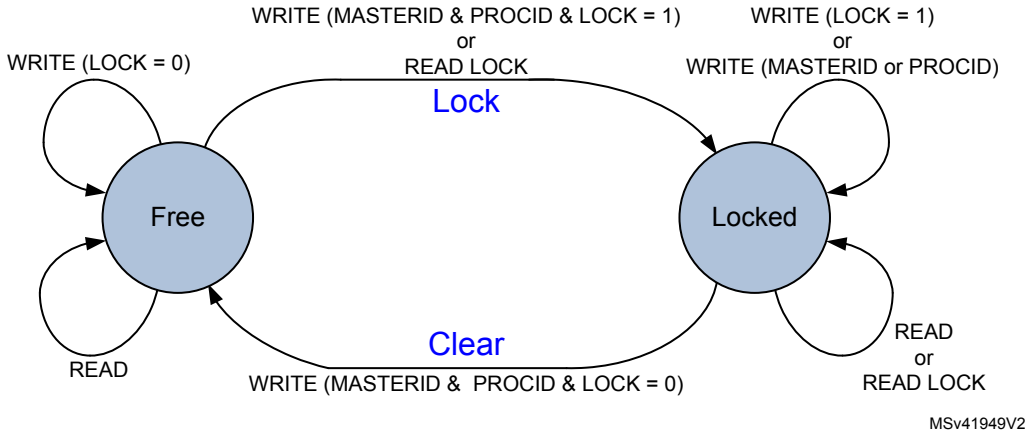
SR5 E1 microcontrollers include a memory-mapped semaphore module (HSEM2) that ensures synchronization among different processes running between the two Arm® Cortex®-M7 cores.

The HSEM2 module must be used only if the cores are in split mode.

The following functions are provided:

- Two locking mechanisms:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check.
  - 1-step lock: by reading the COREID from the semaphore.
- Interrupt generation when a semaphore is freed; a dedicated interrupt is provided for each core.
- Semaphore clear protection.

Figure 3. SR5 E1 semaphore state diagram



A semaphore can only be locked when it is free. Clearing a semaphore is a protected process, to prevent accidental clearing by an AHB bus controller or by a process that does not have the semaphore lock right. The semaphore clear procedure consists in writing to the semaphore with the corresponding COREID and PROCID and LOCK = 0. When the semaphore is cleared, if enabled, an interrupt may be generated to signal the event.

An example of multicore synchronization using the hardware semaphore is shown in the code below, in which the two cores are supposed to be configured in split mode with Core 2 not enabled by hardware.

```
main_core1
…
    /* Take the semaphore 30.*/
    (void)hsem_take_sem_2s(&DRV_HSEM, HSEM_SEM_ID_30, 0U);

    /* Run core2.*/
    RCC->C2_VTOR_INIT_REG  = 0x080F0000UL;
    RCC->C2_BOOT_CTRL_REG |= (RCC_C2_BOOT_CTRL_REG_C2_RES_RELEASE);
    RCC->C2_BOOT_CTRL_REG |= (RCC_C2_BOOT_CTRL_REG_C2_CPU_WAIT_RELEASE);

    /* Wait 2 seconds.*/
    osal_delay_sec(2U);

    /* Release the semaphore 30.*/
    hsem_release_sem(&DRV_HSEM, HSEM_SEM_ID_30, 0U);
…

main_core2
…
    /* Wait till the sempaghore 30 is unlocked.*/
    while (hsem_is_sem_taken(&DRV_HSEM, HSEM_SEM_ID_30) == true);
…
```

Core 1 takes the semaphore 30, then will start Core 2. Core 2 will wait for the semaphore 30 is freed by Core 1 (after 2 seconds), then it will continue its execution.

In addition to the semaphores, each core can generate an interrupt or trigger a wake-up to the other core through the extended interrupt and event controller (EXTI) module. Cortex-M7 core features an output signal as a result of the send event (SEV) instruction. The Core1_SEV is connected to Core 2 NVIC (NVIC2), while Core2_SEV is connected to Core 1 NVIC (NVIC1). So, a SEV instruction executed on Core 1 generates an interrupt on Core 2, and vice versa.

Appendix A Reference documents

Table 3. Reference documents

Name	Title
DS13808	R5 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, and ASIL D
RM0483	SR5E1x 32-bit Arm® Cortex®-M7 architecture microcontroller for electrical vehicle applications

Revision history

Table 4. **Document revision history**

Date	Revision	Changes
24-Jan-2024	1	First release.

Contents

List of tables

List of figures