2 System and memory overview

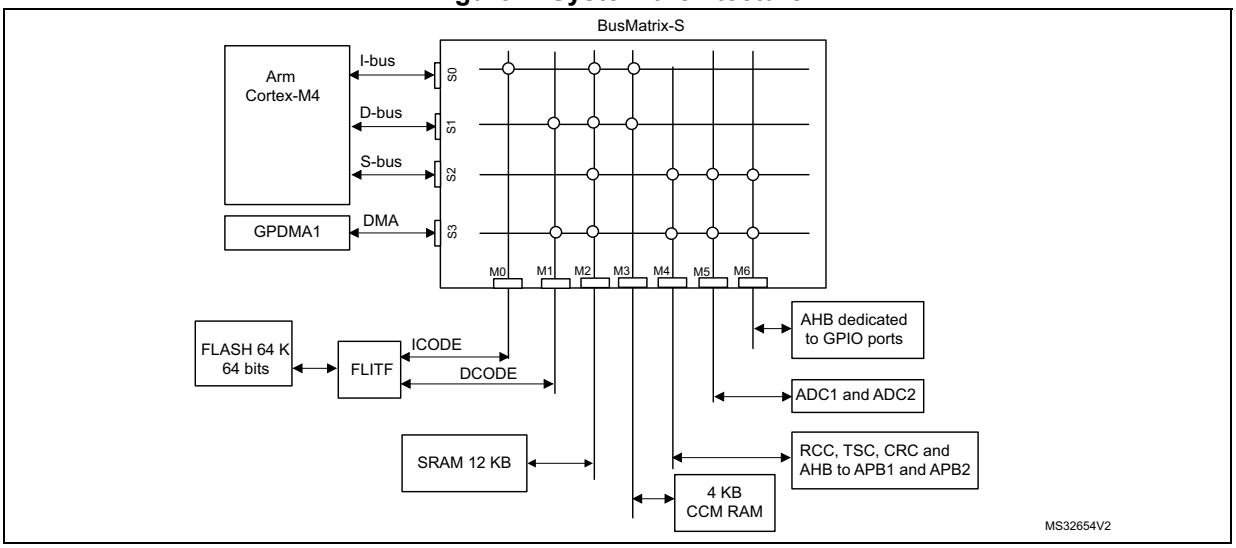
2.1 System architecture

The STM32F334xx main system consists of:

- Four masters:
  - Cortex®-M4 core I-bus
  - Cortex®-M4 core D-bus
  - Cortex®-M4 core S-bus
  - DMA1 (general-purpose DMA)
- Seven slaves:
  - Internal Flash memory on the DCode
  - Internal Flash memory on ICode
  - Up to Internal 12-Kbyte SRAM
  - Internal 4-Kbyte CCM SRAM
  - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
  - AHB dedicated to GPIO ports
  - ADCs 1 and 2

These are interconnected using a multilayer AHB bus architecture as shown in *Figure 1*:

Figure 1. System architecture



2.1.1 S0: I-bus

This bus connects the Instruction bus of the Cortex®-M4 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory, the SRAM up to 16 Kbytes and the CCM SRAM (4 Kbytes).

2.1.2 S1: D-bus

This bus connects the DCode bus (literal load and debug access) of the Cortex®-M4 core to the BusMatrix. The targets of this bus are the internal Flash memory, the SRAM (16 Kbytes) and the CCM SRAM (4 Kbytes).

2.1.3 S2: S-bus

This bus connects the system bus of the Cortex®-M4 core to the BusMatrix. This bus is used to access data located in the peripheral or SRAM area. The targets of this bus are the SRAM (16 Kbytes), the AHB to APB1/APB2 bridges, the AHB IO port and the 2 ADCs.

2.1.4 S3: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of different Masters to Flash, SRAM (16 Kbytes) and peripherals.

2.1.5 BusMatrix

The BusMatrix manages the access arbitration between Masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed of five masters (CPU AHB, System bus, DCode bus, ICode bus, DMA1/2 bus) and seven slaves (FLITF, SRAM, CCM SRAM, AHB2GPIO and AHB2APB1/2 bridges, and ADCs).

AHB/APB bridges

The two AHB/APB bridges provide full synchronous connections between the AHB and the two APB buses. APB1 is limited to 36 MHz, APB2 operates at full speed (72 MHz).

Refer to *Section 2.2.2: Memory map and register boundary addresses on page 48* for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and FLITF). Before using a peripheral user has to enable its clock in the RCC_AHBENR, RCC_APB2ENR or RCC_APB1ENR register.

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

2.2 Memory organization

2.2.1 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.

The addressable memory space is divided into eight main blocks, of 512 Mbytes each.

2.2.2 Memory map and register boundary addresses

Refer to the device datasheet for a comprehensive diagram of the memory map.

The following table gives the boundary addresses of the peripherals available in the devices.

Table 1. STM32F334xx peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5000 0000 - 0x5000 03FF	1 K	ADC1 - ADC2	*Section 13.7 on page 313*
-	0x4800 1800 - 0x4FFF FFFF	~132 M	Reserved	-
AHB2	0x4800 1400 - 0x4800 17FF	1 K	GPIOF	*Section 9.4.12 on page 155*
	0x4800 1000 - 0x4800 13FF	1 K	Reserved	
	0x4800 0C00 - 0x4800 0FFF	1 K	GPIOD	
	0x4800 0800 - 0x4800 0BFF	1 K	GPIOC	
	0x4800 0400 - 0x4800 07FF	1 K	GPIOB	
	0x4800 0000 - 0x4800 03FF	1 K	GPIOA	
-	0x4002 4400 - 0x47FF FFFF	~128 M	Reserved	
AHB1	0x4002 4000 - 0x4002 43FF	1 K	TSC	*Section 17.6.11 on page 381*
	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 K	CRC	*Section 5.4.6 on page 81*
	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 K	Flash interface	*Section 3.6 on page 71*
	0x4002 1400 - 0x4002 1FFF	3 K	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 K	RCC	*Section 8.4.14 on page 137*
	0x4002 0400 - 0x4002 0FFF	3 K	Reserved	-
	0x4002 0000 - 0x4002 03FF	1 K	DMA1	*Section 11.6.7 on page 190*
-	0x4001 8000 - 0x4001 FFFF	32 K	Reserved	-

Table 1. STM32F334xx peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x4001 7400 - 0x4001 77FF	1 K	HRTIM1	*Section 21.5.64 on page 796*
	0x4001 4C00 - 0x4001 73FF	12 K	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*Section 20.6.18 on page 625*
	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x4001 4000 - 0x4001 43FF	1 K	TIM15	*Section 20.5.19 on page 605*
	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*Section 25.7.12 on page 708*
	0x4001 3400 - 0x4001 37FF	1 K	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 K	SPI1	*Section 29.6.8 on page 1049*
	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	*Section 18.4.27 on page 475*
	0x4001 0800 - 0x4001 2BFF	9 K	Reserved	-
	0x4001 0400 - 0x4001 07FF	1 K	EXTI	*Section 12.3.13 on page 208*
	0x4001 0000 - 0x4001 03FF	1 K	SYSCFG + COMP + OPAMP	*Section 10.1.9 on page 168*, *Section 15.5.4 on page 352*, *Section 16.4.2 on page 363*
-	0x4000 9C00 - 0x4000 FFFF	25 K	Reserved	-

Table 1. STM32F334xx peripheral register boundary addresses (continued)

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1	0x4000 9800 - 0x4000 9BFF	1 K	DAC2	*Section 14.10.15 on page 341*
	0x4000 7800 - 0x4000 97FF	8 K	Reserved	-
	0x4000 7400 - 0x4000 77FF	1 K	DAC1	*Section 14.10.15 on page 341*
	0x4000 7000 - 0x4000 73FF	1 K	PWR	*Section 6.4.3 on page 95*
	0x4000 6800 - 0x4000 6FFF	2 K	Reserved	-
	0x4000 6400 - 0x4000 67FF	1 K	bxCAN	*Section 30.9.5 on page 1090*
	0x4000 5800 - 0x4000 63FF	3 K	Reserved	-
	0x4000 5400 - 0x4000 57FF	1 K	I2C1	*Section 27.7.12 on page 946*
	0x4000 4C00 - 0x4000 53FF	2 K	Reserved	-
	0x4000 4800 - 0x4000 4BFF	1 K	USART3	*Section 25.7.12 on page 708*
	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x4000 3400 - 0x4000 43FF	4 K	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*Section 24.4.6 on page 827*
	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	*Section 25.5.4 on page 833*
	0x4000 2800 - 0x4000 2BFF	1 K	RTC	*Section 26.6.20 on page 875*
	0x4000 1800 - 0x4000 27FF	4 K	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 K	TIM7	*Section 23.4.9 on page 818*
	0x4000 1000 - 0x4000 13FF	1 K	TIM6	*Section 23.4.9 on page 818*
	0x4000 0800 - 0x4000 0FFF	2 K	Reserved	-
	0x4000 0400 - 0x4000 07FF	1 K	TIM3	*Section 19.4.22 on page 546*
	0x4000 0000 - 0x4000 03FF	1 K	TIM2	
-	0x2000 A000 - 3FFF FFFF	~512 M	Reserved	-
-	0x2000 0000 - 0x2000 2FFF	12 K	SRAM	-
-	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	-
-	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	-
-	0x1000 2000 - 0x1FFF D7FF	~256 M	Reserved	-
-	0x1000 0000 - 0x1000 0FFF	4 K	CCM SRAM	-
-	0x0804 0000 - 0x0FFF FFFF	~128 M	Reserved	-
-	0x0800 0000 - 0x0800 FFFF	64 K	Main Flash memory	-
-	0x0004 0000 - 0x07FF FFFF	~128 M	Reserved	-
-	0x0000 000 - 0x0000 FFFF	64 K	Main Flash memory, system memory or SRAM depending on BOOT configuration	-

2.2.3 Parity check

The parity check is implemented on all of the SRAM and CCM SRAM. The SRAM parity check is disabled by default. It is enabled by the user, when needed, using an option bit.

The data bus width of the SRAM supporting the parity check is 36 bits because 4 bits are available for parity check (1 bit per byte) in order to increase memory robustness, as required for instance by Class B or SIL norms.

The parity bits are computed on data and address and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated if the SRAM parity check is enabled. The same error can also be linked to the Break input of TIMER 1, 8, 15, 16 and 17, by setting the SRAM_PARITY_LOCK control bit in the SYSCFG configuration register 2 (SYSCFG_CFGR2). In case of parity error, the SRAM Parity Error flag (SRAM_PEF) is set in SYSCFG_CFGR2. For more details, refer to SYSCFG_CFGR2.

The BYP_ADD_PAR bit in SYSCFG_CFGR2 can be used to prevent an unwanted parity error to occur when the user programs a code in the RAM at address 0x2XXXXXXX (address in the address range 0x20000000-0x20002000) and then executes the code from RAM at boot (RAM is remapped at address 0x00).

2.2.4 CCM SRAM write protection

The CCM SRAM is write protected with a page granularity of 1 Kbyte.

Table 2. CCM SRAM organization

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF

The write protection can be enabled in the CCM SRAM protection register (SYSCFG_RCR) in the SYSCFG block. This is a register with write 1 once mechanism, which means by writing 1 on a bit, it sets up the write protection for that page of SRAM and it can be removed/cleared by a system reset only. For more details, refer to the SYSCFG section.

2.3 Embedded SRAM

STM32F334xx devices feature up to 16 Kbytes of static SRAM. It can be accessed as bytes, halfwords (16 bits) or full words (32 bits):

- Up to 12 Kbytes of SRAM that can be addressed at maximum system clock frequency
- without wait states and can be accessed by both CPU and DMA;
- 4 Kbytes of CCM SRAM. It is used to execute critical routines or to access data. It can be accessed by the CPU only. No DMA accesses are allowed. This memory can be addressed at maximum system clock frequency without wait state.

2.4 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of two parts:
  - Option bytes for hardware and memory protection user configuration
  - System memory which contains the proprietary boot loader code. Refer to *Section 3: Embedded Flash memory* for more details.

Flash memory instructions and data access are performed through the AHB bus. The prefetch block is used for instruction fetches through the ICode bus. Arbitration is performed in the Flash memory interface, and priority is given to data access on the DCode bus. It also implements the logic necessary to carry out the Flash memory operations (Program/Erase) controlled through the Flash registers.

2.5 Boot configuration

In the STM32F334xx, three different boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the user option byte, as shown in the following table:

Table 3. Boot modes

Boot mode selection		Boot mode	Aliasing
nBOOT1	BOOT0	-	-
x	0	Main Flash memory	Main flash memory selected as boot area
1	1	System memory	System memory selected as boot area
0	1	Embedded SRAM	Embedded SRAM (on the DCode bus) selected as boot area

The values on both BOOT0 pin and nBOOT1 bit are latched on the 4th rising edge of SYSCLK after a reset.

It is up to the user to set the nBOOT1 and BOOT0 to select the required boot mode. The BOOT0 pin and nBOOT1 bit are also resampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004. Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF D800).
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

2.5.1 Embedded boot loader

The embedded boot loader is located in the system memory, programmed by ST during production. It is used to reprogram the Flash memory through USART1(PA9/PA10), USART2(PA2/PA3) or I2C1(PB6/PB7).