2 Memory and bus architecture

2.1 System architecture

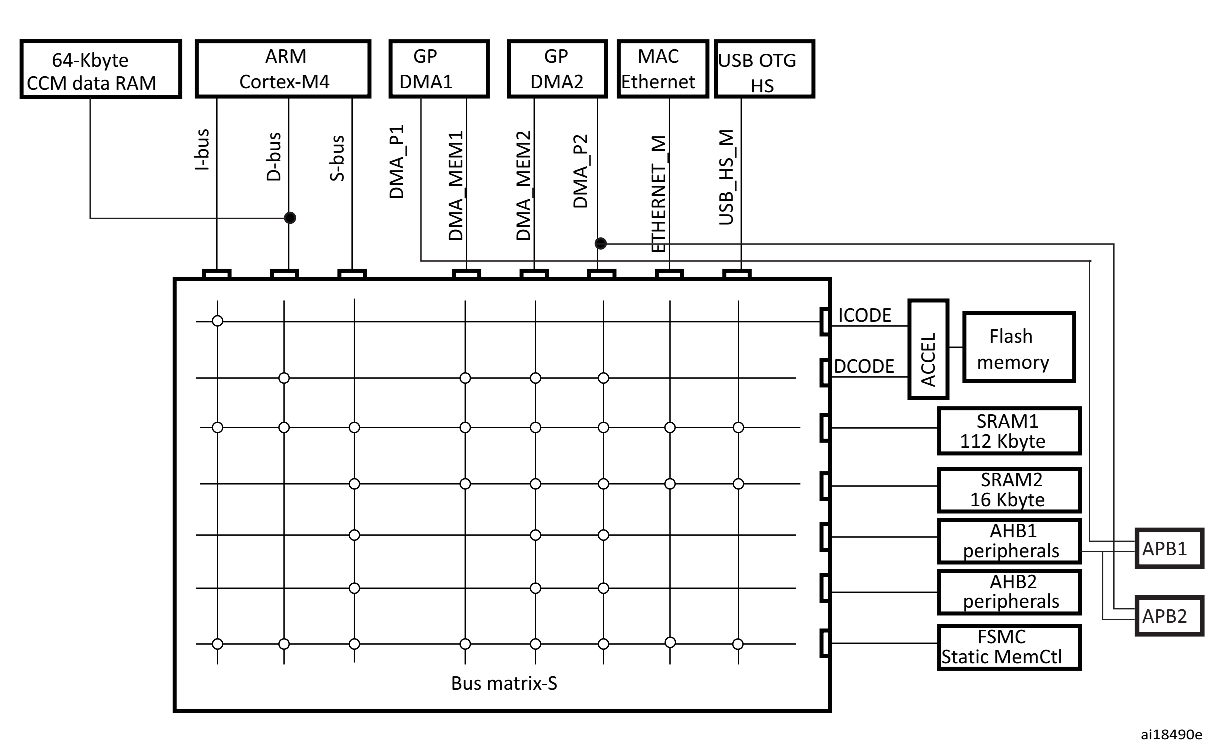
In STM32F405xx/07xx and STM32F415xx/17xx, the main system consists of 32-bit multilayer AHB bus matrix that interconnects:

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Eight masters:
  - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
  - DMA2 peripheral bus
  - Ethernet DMA bus
  - USB OTG HS DMA bus
- Seven slaves:
  - Internal flash memory ICode bus
  - Internal flash memory DCode bus
  - Main internal SRAM1 (112 KB)
  - Auxiliary internal SRAM2 (16 KB)
  - AHB1 peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals
  - FSMC

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. The 64-Kbyte CCM (core coupled memory) data RAM is not part of the bus matrix and can be accessed only through the CPU. This architecture is shown in *Figure 1*.

Figure 1. System architecture for STM32F405xx/07xx and STM32F415xx/17xx devices

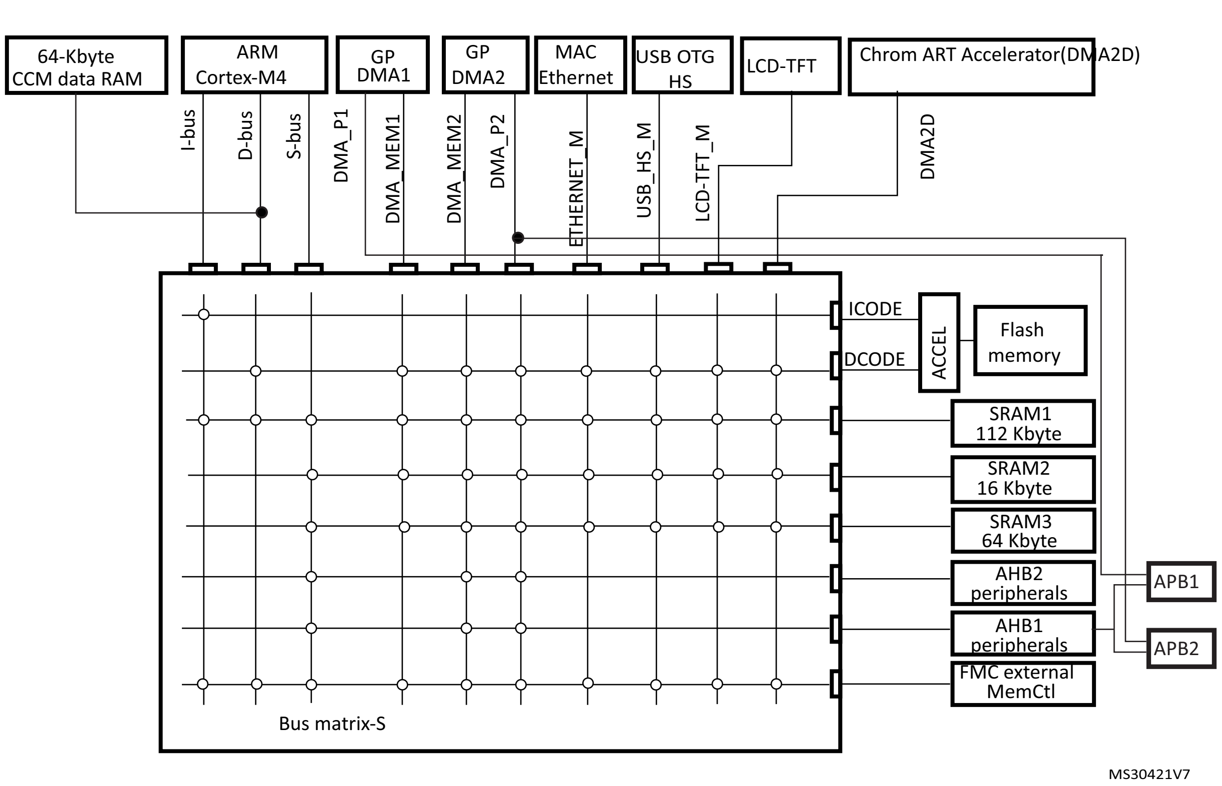


In the STM32F42xx and STM32F43xx devices, the main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- Ten masters:
  - Cortex®-M4 with FPU core I-bus, D-bus and S-bus
  - DMA1 memory bus
  - DMA2 memory bus
  - DMA2 peripheral bus
  - Ethernet DMA bus
  - USB OTG HS DMA bus
  - LCD Controller DMA-bus
  - DMA2D (Chrom-Art Accelerator™) memory bus
- Eight slaves:
  - Internal flash memory ICode bus
  - Internal flash memory DCode bus
  - Main internal SRAM1 (112 KB)
  - Auxiliary internal SRAM2 (16 KB)
  - Auxiliary internal SRAM3 (64 KB)
  - AHB1peripherals including AHB to APB bridges and APB peripherals
  - AHB2 peripherals
  - FMC

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. The 64-Kbyte CCM (core coupled memory) data RAM is not part of the bus matrix and can be accessed only through the CPU. This architecture is shown in *Figure 2*.

Figure 2. System architecture for STM32F42xxx and STM32F43xxx devices



2.1.1 I-bus

This bus connects the Instruction bus of the Cortex®-M4 with FPU core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (internal flash memory/SRAM or external memories through the FSMC/FMC).

2.1.2 D-bus

This bus connects the databus of the Cortex®-M4 with FPU to the 64-Kbyte CCM data RAM to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code or data (internal flash memory or external memories through the FSMC/FMC).

2.1.3 S-bus

This bus connects the system bus of the Cortex®-M4 with FPU core to a BusMatrix. This bus is used to access data located in a peripheral or in SRAM. Instructions may also be fetched on this bus (less efficient than ICode). The targets of this bus are the internal SRAM1, SRAM2 and SRAM3, the AHB1 peripherals including the APB peripherals, the AHB2 peripherals and the external memories through the FSMC/FMC.

2.1.4 DMA memory bus

This bus connects the DMA memory bus master interface to the BusMatrix. It is used by the DMA to perform transfer to/from memories. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2 and SRAM3) and external memories through the FSMC/FMC.

2.1.5 DMA peripheral bus

This bus connects the DMA peripheral master bus interface to the BusMatrix. This bus is used by the DMA to access AHB peripherals or to perform memory-to-memory transfers. The targets of this bus are the AHB and APB peripherals plus data memories: internal SRAMs (SRAM1, SRAM2 and SRAM3) and external memories through the FSMC/FMC.

2.1.6 Ethernet DMA bus

This bus connects the Ethernet DMA master interface to the BusMatrix. This bus is used by the Ethernet DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal flash memory, and external memories through the FSMC/FMC.

2.1.7 USB OTG HS DMA bus

This bus connects the USB OTG HS DMA master interface to the BusMatrix. This bus is used by the USB OTG DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), internal flash memory, and external memories through the FSMC/FMC.

2.1.8 LCD-TFT controller DMA bus

This bus connects the LCD controller DMA master interface to the BusMatrix. It is used by the LCD-TFT DMA to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), external memories through FMC, and internal flash memory.

2.1.9 DMA2D bus

This bus connect the DMA2D master interface to the BusMatrix. This bus is used by the DMA2D graphic Accelerator to load/store data to a memory. The targets of this bus are data memories: internal SRAMs (SRAM1, SRAM2, SRAM3), external memories through FMC, and internal flash memory.

2.1.10 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm.

2.1.11 AHB/APB bridges (APB)

The two AHB/APB bridges, APB1 and APB2, provide full synchronous connections between the AHB and the two APB buses, allowing flexible selection of the peripheral frequency.

Refer to the device datasheets for more details on APB1 and APB2 maximum frequencies, and to *Table 1* for the address mapping of AHB and APB peripherals.

After each device reset, all peripheral clocks are disabled (except for the SRAM and flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR or RCC_APBxENR register.

Note: When a 16- or an 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

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 word’s most significant.

For the detailed mapping of peripheral registers, please refer to the related chapters.

The addressable memory space is divided into 8 main blocks, each of 512 MB.

All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”). Refer to the memory map figure in the product datasheet.

2.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 1* gives the boundary addresses of the peripherals available in all STM32F4xx devices.

Table 1. STM32F4xx register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC control register (STM32F405xx/07xx and STM32F415xx/17xx)/ FMC control register (STM32F42xxx and STM32F43xxx)	AHB3	*Section 36.6.9: FSMC register map on page 1603* *Section 37.8: FMC register map on page 1683*

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x5006 0800 - 0x5006 0BFF	RNG	AHB2	*Section 24.4.4: RNG register map on page 774*
0x5006 0400 - 0x5006 07FF	HASH		*Section 25.4.9: HASH register map on page 798*
0x5006 0000 - 0x5006 03FF	CRYP		*Section 23.6.13: CRYP register map on page 766*
0x5005 0000 - 0x5005 03FF	DCMI		*Section 15.8.12: DCMI register map on page 481*
0x5000 0000 - 0x5003 FFFF	USB OTG FS		*Section 34.16.6: OTG_FS register map on page 1329*
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1	*Section 35.12.6: OTG_HS register map on page 1475*
0x4002 B000 - 0x4002 BBFF	DMA2D		*Section 11.5: DMA2D registers on page 355*
0x4002 8000 - 0x4002 93FF	ETHERNET MAC		*Section 33.8.5: Ethernet register maps on page 1239*
0x4002 6400 - 0x4002 67FF	DMA2		*Section 10.5.11: DMA register map on page 338*
0x4002 6000 - 0x4002 63FF	DMA1		
0x4002 4000 - 0x4002 4FFF	BKPSRAM		-
0x4002 3C00 - 0x4002 3FFF	Flash interface register		*Section 3.9: Flash interface registers*
0x4002 3800 - 0x4002 3BFF	RCC		*Section 7.3.24: RCC register map on page 267*
0x4002 3000 - 0x4002 33FF	CRC		*Section 4.4.4: CRC register map on page 116*
0x4002 2800 - 0x4002 2BFF	GPIOK		*Section 8.4.11: GPIO register map on page 290*
0x4002 2400 - 0x4002 27FF	GPIOJ		
0x4002 2000 - 0x4002 23FF	GPIOI		*Section 8.4.11: GPIO register map on page 290*
0x4002 1C00 - 0x4002 1FFF	GPIOH		
0x4002 1800 - 0x4002 1BFF	GPIOG		
0x4002 1400 - 0x4002 17FF	GPIOF		
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		
0x4001 6800 - 0x4001 6BFF	LCD-TFT	APB2	*Section 16.7.26: LTDC register map on page 515*
0x4001 5800 - 0x4001 5BFF	SAI1		*Section 29.17.9: SAI register map on page 966*
0x4001 5400 - 0x4001 57FF	SPI6	APB2	*Section 28.5.10: SPI register map on page 928*
0x4001 5000 - 0x4001 53FF	SPI5		

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	Section 19.5.12: TIM10/11/13/14 register map on page 697
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		Section 19.4.13: TIM9/12 register map on page 687
0x4001 3C00 - 0x4001 3FFF	EXTI		Section 12.3.7: EXTI register map on page 390
0x4001 3800 - 0x4001 3BFF	SYSCFG		Section 9.2.8: SYSCFG register maps for STM32F405xx/07xx and STM32F415xx/17xx on page 297 and Section 9.3.8: SYSCFG register maps for STM32F42xxx and STM32F43xxx on page 304
0x4001 3400 - 0x4001 37FF	SPI4	APB2	Section 28.5.10: SPI register map on page 928
0x4001 3000 - 0x4001 33FF	SPI1	APB2	Section 28.5.10: SPI register map on page 928
0x4001 2C00 - 0x4001 2FFF	SDIO		Section 31.9.16: SDIO register map on page 1077
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		Section 13.13.18: ADC register map on page 433
0x4001 1400 - 0x4001 17FF	USART6		Section 30.6.8: USART register map on page 1021
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		Section 17.4.21: TIM1 and TIM8 register map on page 590
0x4001 0000 - 0x4001 03FF	TIM1		
0x4000 7C00 - 0x4000 7FFF	UART8	APB1	Section 30.6.8: USART register map on page 1021
0x4000 7800 - 0x4000 7BFF	UART7		

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC	APB1	Section 14.5.15: DAC register map on page 456
0x4000 7000 - 0x4000 73FF	PWR		Section 5.6: PWR register map on page 151
0x4000 6800 - 0x4000 6BFF	CAN2		Section 32.9.5: bxCAN register map on page 1121
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		Section 27.6.11: I2C register map on page 875
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		Section 30.6.8: USART register map on page 1021
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		Section 28.5.10: SPI register map on page 928
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		Section 21.4.5: IWDG register map on page 715
0x4000 2C00 - 0x4000 2FFF	WWDG		Section 22.6.4: WWDG register map on page 722
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		Section 26.6.21: RTC register map on page 839
0x4000 2000 - 0x4000 23FF	TIM14		Section 19.5.12: TIM10/11/13/14 register map on page 697
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		Section 19.4.13: TIM9/12 register map on page 687
0x4000 1400 - 0x4000 17FF	TIM7		Section 20.4.9: TIM6 and TIM7 register map on page 710
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5		Section 18.4.21: TIMx register map on page 651
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

2.3.1 Embedded SRAM

The STM32F405xx/07xx and STM32F415xx/17xx feature 4 Kbytes of backup SRAM (see *Section 5.1.2: Battery backup domain*) plus 192 Kbytes of system SRAM.

The STM32F42xxx and STM32F43xxx feature 4 Kbytes of backup SRAM (see *Section 5.1.2: Battery backup domain*) plus 256 Kbytes of system SRAM.

The embedded SRAM can be accessed as bytes, half-words (16 bits) or full words (32 bits). Read and write operations are performed at CPU speed with 0 wait state. The embedded SRAM is divided into up to three blocks:

- SRAM1 and SRAM2 mapped at address 0x2000 0000 and accessible by all AHB masters.
- SRAM3 (available on STM32F42xxx and STM32F43xxx) mapped at address 0x2002 0000 and accessible by all AHB masters.
- CCM (core coupled memory) mapped at address 0x1000 0000 and accessible only by the CPU through the D-bus.

The AHB masters support concurrent SRAM accesses (from the Ethernet or the USB OTG HS): for instance, the Ethernet MAC can read/write from/to SRAM2 while the CPU is reading/writing from/to SRAM1 or SRAM3.

The CPU can access the SRAM1 through the System bus or through the I-Code/D-Code buses when boot from SRAM is selected or when physical remap is selected (*Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller). To get the max performance on SRAM execution, physical remap should be selected (boot or software selection).

2.3.2 Flash memory overview

The flash memory interface manages CPU AHB I-Code and D-Code accesses to the flash memory. It implements the erase and program flash memory operations and the read and write protection mechanisms. It accelerates code execution with a system of instruction prefetch and cache lines.

The flash memory is organized as follows:

- A main memory block divided into sectors.
- System memory from which the device boots in System memory boot mode
- 512 OTP (one-time programmable) bytes for user data.
- Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

Refer to *Section 3: Embedded flash memory interface* for more details.

2.3.3 Bit banding

The Cortex®-M4 with FPU memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the STM32F4xx devices both the peripheral registers and the SRAM are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for Cortex®-M4 with FPU accesses, and not from other bus masters (e.g. DMA).

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

bit_word_addr = *bit_band_base* + (*byte_offset* x 32) + (*bit_number* × 4)

where:

- *bit_word_addr* is the address of the word in the alias memory region that maps to the targeted bit
- *bit_band_base* is the starting address of the alias region
- *byte_offset* is the number of the byte in the bit-band region that contains the targeted bit
- *bit_number* is the bit position (0-7) of the targeted bit

Example

The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 to the alias region:

0x22006008 = 0x22000000 + (0x300*32) + (2*4)

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset).

For more information on bit-banding, please refer to the *Cortex®*-M4 with FPU *programming manual* (see *Related documents on page 1*).

2.4 Boot configuration

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex®-M4 with FPU CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, flash memory). STM32F4xx microcontrollers implement a special mechanism to be able to boot from other memories (like the internal SRAM).

In the STM32F4xx, three different boot modes can be selected through the BOOT[1:0] pins as shown in *Table 2*.

Table 2. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main flash memory	Main flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after reset to select the required boot mode.

BOOT0 is a dedicated pin while BOOT1 is shared with a GPIO pin. Once BOOT1 has been sampled, the corresponding GPIO pin is free and can be used for other purposes.

The BOOT pins are also resampled when the device exits the Standby mode. Consequently, they must be kept in the required Boot mode configuration when the device is in the Standby mode. After this startup delay is over, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.

Note: When the device boots from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and the offset register.

In STM32F42xxx and STM32F43xxx devices, when booting from the main flash memory, the application software can either boot from bank 1 or from bank 2. By default, boot from bank 1 is selected.

To select boot from flash memory bank 2, set the BFB2 bit in the user option bytes. When this bit is set and the boot pins are in the boot from main flash memory configuration, the device boots from system memory, and the boot loader jumps to execute the user application programmed in flash memory bank 2. For further details, please refer to AN2606.

Embedded bootloader

The embedded bootloader mode is used to reprogram the flash memory using one of the following serial interfaces:

- USART1 (PA9/PA10)
- USART3 (PB10/11 and PC10/11)
- CAN2 (PB5/13)
- USB OTG FS (PA11/12) in Device mode (DFU: device firmware upgrade).

The USART peripherals operate at the internal 16 MHz oscillator (HSI) frequency, while the CAN and USB OTG FS require an external clock (HSE) multiple of 1 MHz (ranging from 4 to 26 MHz).

The embedded bootloader code is located in system memory. It is programmed by ST during production. For additional information, refer to application note AN2606.

Physical remap in STM32F405xx/07xx and STM32F415xx/17xx

Once the boot pins are selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the *Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memories can thus be remapped:

- Main flash memory
- System memory
- Embedded SRAM1 (112 KB)
- FSMC bank 1 (NOR/PSRAM 1 and 2)

Table 3. Memory mapping vs. Boot mode/physical remap in STM32F405xx/07xx and STM32F415xx/17xx

Addresses	Boot/Remap in main flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FSMC
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x080F FFFF	Flash memory	Flash memory	Flash memory	Flash memory
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FSMC bank 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x000F FFFF(1)(2)	Flash (1 MB) Aliased	SRAM1 (112 KB) Aliased	System memory (30 KB) Aliased	FSMC bank 1 NOR/PSRAM 1 (128 MB Aliased)

1. When the FSMC is remapped at address 0x0000 0000, only the first two regions of bank 1 memory controller (bank 1 NOR/PSRAM 1 and NOR/PSRAM 2) can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.
2. Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

Physical remap in STM32F42xxx and STM32F43xxx

Once the boot pins are selected, the application software can modify the memory accessible in the code area (in this way the code can be executed through the ICode bus in place of the System bus). This modification is performed by programming the *Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memories can thus be remapped:

- Main flash memory
- System memory
- Embedded SRAM1 (112 KB)
- FMC bank 1 (NOR/PSRAM 1 and 2)
- FMC SDRAM bank 1

Table 4. Memory mapping vs. Boot mode/physical remap in STM32F42xxx and STM32F43xxx

Addresses	Boot/Remap in main flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FMC
0x2002 0000 - 0x2002 FFFF	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	System memory	System memory	System memory	System memory
0x0810 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved	Reserved
0x0800 0000 - 0x081F FFFF	Flash memory	Flash memory	Flash memory	Flash memory

Table 4. Memory mapping vs. Boot mode/physical remap in STM32F42xxx and STM32F43xxx (continued)

Addresses	Boot/Remap in main flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory	Remap in FMC
0x0400 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved	FMC bank 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x001F FFFF(1)(2)	Flash (2 MB) Aliased	SRAM1 (112 KB) Aliased	System memory (30 KB) Aliased	FMC bank 1 NOR/PSRAM 1 (128 MB Aliased) or FMC SDRAM bank 1 (128 MB Aliased)

1. When the FMC is remapped at address 0x0000 0000, only the first two regions of bank 1 memory controller (bank 1 NOR/PSRAM 1 and NOR/PSRAM 2) or SDRAM bank 1 can be remapped. In remap mode, the CPU can access the external memory via ICode bus instead of System bus which boosts up the performance.
2. Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.