

Understanding STM32 Memory Architecture



SRAM

- Used for runtime data: variables, stack, heap
- Fast read/write access
- Limited in size (e.g., 20KB–256KB on many STM32 MCUs)
- Volatile memory (data lost on power-off)

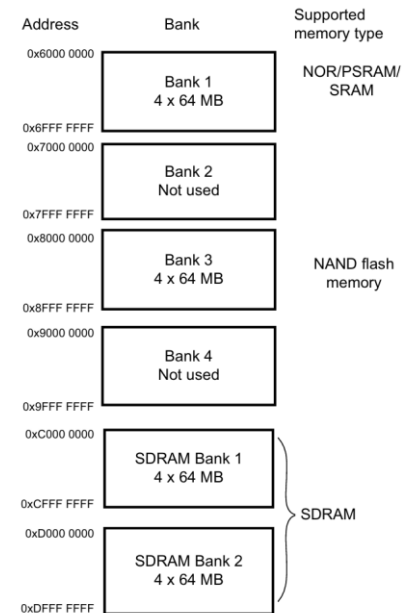
Flash Memory

- Non-volatile memory (retains data after power-off)
- Stores program code and constant data
- Slower than SRAM, read-only during runtime
- Requires erasing blocks before writing
- Flash is structured in pages and blocks:
 - You can only write in pages
 - You can only erase in blocks

NOR vs NAND FLASH

Feature	NOR Flash	NAND Flash
Access Type	Random access (like RAM)	Block access (like a file system)
Read Speed	Fast	Faster for sequential reads
Write/Erase	Slower	Faster and more efficient
Cost per Bit	Higher	Lower
Endurance	Higher	Lower

- The on board flash of STM32F7 is NOR-based
- The external flash can be nor or nand; For each bank the type of memory to be used can be configured by the user application through the Configuration register.



Memory-Mapped Memory

- Each memory region (Flash, SRAM, peripherals) has a unique address
- Code and data are accessed using memory addresses
- Enables direct access to hardware through normal pointers

Memory-Mapped I/O

- Peripherals are accessible via memory addresses
- Example GPIO register at 0x48000000
- Read/write operations to control hardware
- Efficient and flexible hardware interaction

Registers

- Small storage locations in peripherals
- Control hardware behavior (e.g., GPIO mode, ADC settings)
- Accessed via memory-mapped I/O
- Bit-level control for fine-grained configuration

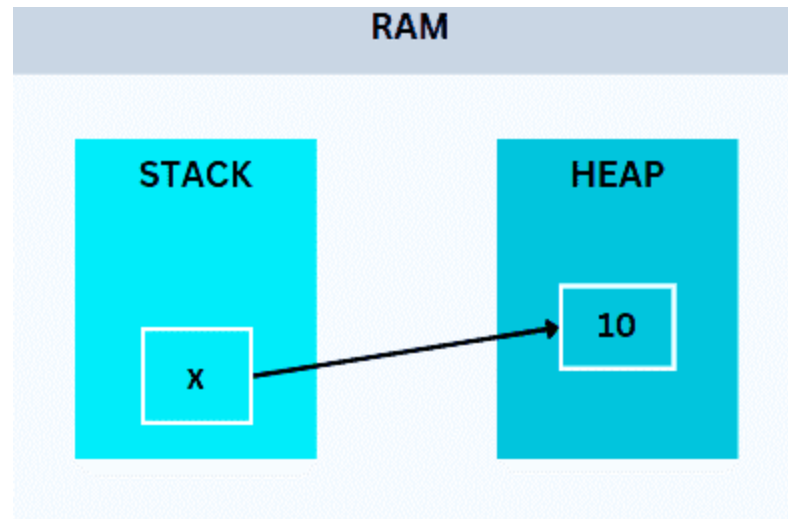
2.2.2 Memory map and register boundary addresses

0xFFFF FFFF	512-Mbyte Block 7 Cortex-M7 Internal peripherals		Reserved	0xE010 0000 - 0xFFFF FFFF																											
0xE000 0000	512-Mbyte Block 6 FMC		Cortex-M7 internal peripherals	0xE000 0000 - 0xE00F FFFF																											
0xD000 0000			AHB3	0x6000 0000 - 0xDFFF FFFF																											
0xC000 0000	512-Mbyte Block 5 FMC		Reserved	0x5006 0C00 - 0x5FFF FFFF																											
0x9FFF FFFF			AHB2	0x5006 0BFF																											
0x8000 0000	512-Mbyte Block 4 Quad-SPI and FMC bank 3		Reserved	0x5000 0000																											
0x7FFF FFFF			AHB1	0x4008 0000 - 0x4FFF FFFF																											
0x6000 0000	512-Mbyte Block 3 FMC bank 1 to bank 2		Reserved	0x4007 FFFF																											
0x5FFF FFFF			Reserved	0x4002 0000																											
0x4000 0000	512-Mbyte Block 2 Peripherals		Reserved	0x4001 6C00 - 0x4001 FFFF																											
0x3FFF FFFF			APB2	0x4001 6BFF																											
0x2000 0000	512-Mbyte Block 1 SRAM		Reserved	0x4001 0000																											
0x1FFF FFFF			Reserved	0x4000 8000 - 0x4000 FFFF																											
0x0000 0000	<table border="1"> <tr><td>Reserved</td><td>0x2008 0000 - 0x3FFF FFFF</td></tr> <tr><td>SRAM2 (16 KB)</td><td>0x2007 C000 - 0x2007 FFFF</td></tr> <tr><td>SRAM1 (368 KB)</td><td>0x2002 0000 - 0x2007 BFFF</td></tr> <tr><td>DTCM (128 KB)</td><td>0x2000 0000 - 0x2001 FFFF</td></tr> <tr><td>Reserved</td><td>0x1FFF 0020 - 0x1FFF FFFF</td></tr> <tr><td>Option Bytes</td><td>0x1FFF 0000 - 0x1FFF 001F</td></tr> <tr><td>Reserved</td><td>0x0820 0000 - 0x1FFE FFFF</td></tr> <tr><td>Flash memory on AXIM interface</td><td>0x0800 0000 - 0x081F FFFF</td></tr> <tr><td>Reserved</td><td>0x0040 0000 - 0x007F FFFF</td></tr> <tr><td>Flash memory on ITCM interface</td><td>0x0020 0000 - 0x003F FFFF</td></tr> <tr><td>Reserved</td><td>0x0011 0000 - 0x001F FFFF</td></tr> <tr><td>System memory</td><td>0x0010 0000 - 0x0010 FFFF</td></tr> <tr><td>Reserved</td><td>0x0000 4000 - 0x000F FFFF</td></tr> <tr><td>ITCM RAM</td><td>0x0000 0000 - 0x0000 3FFF</td></tr> </table>		Reserved	0x2008 0000 - 0x3FFF FFFF	SRAM2 (16 KB)	0x2007 C000 - 0x2007 FFFF	SRAM1 (368 KB)	0x2002 0000 - 0x2007 BFFF	DTCM (128 KB)	0x2000 0000 - 0x2001 FFFF	Reserved	0x1FFF 0020 - 0x1FFF FFFF	Option Bytes	0x1FFF 0000 - 0x1FFF 001F	Reserved	0x0820 0000 - 0x1FFE FFFF	Flash memory on AXIM interface	0x0800 0000 - 0x081F FFFF	Reserved	0x0040 0000 - 0x007F FFFF	Flash memory on ITCM interface	0x0020 0000 - 0x003F FFFF	Reserved	0x0011 0000 - 0x001F FFFF	System memory	0x0010 0000 - 0x0010 FFFF	Reserved	0x0000 4000 - 0x000F FFFF	ITCM RAM	0x0000 0000 - 0x0000 3FFF	APB1
Reserved		0x2008 0000 - 0x3FFF FFFF																													
SRAM2 (16 KB)		0x2007 C000 - 0x2007 FFFF																													
SRAM1 (368 KB)		0x2002 0000 - 0x2007 BFFF																													
DTCM (128 KB)		0x2000 0000 - 0x2001 FFFF																													
Reserved		0x1FFF 0020 - 0x1FFF FFFF																													
Option Bytes		0x1FFF 0000 - 0x1FFF 001F																													
Reserved		0x0820 0000 - 0x1FFE FFFF																													
Flash memory on AXIM interface		0x0800 0000 - 0x081F FFFF																													
Reserved		0x0040 0000 - 0x007F FFFF																													
Flash memory on ITCM interface	0x0020 0000 - 0x003F FFFF																														
Reserved	0x0011 0000 - 0x001F FFFF																														
System memory	0x0010 0000 - 0x0010 FFFF																														
Reserved	0x0000 4000 - 0x000F FFFF																														
ITCM RAM	0x0000 0000 - 0x0000 3FFF																														
			0x4000 0000																												

MSv39118V4

Stack and Heap

- Stack: stores function calls, local variables
- Heap: used for dynamic memory allocation
- Both reside in SRAM and grow toward each other



TIPS

- The linker file gives you the information about the memories used in the mcu

```
MEMORY
{
  RAM      (xrw)      : ORIGIN = 0x20000000,   LENGTH = 96K
  FLASH    (rx)       : ORIGIN = 0x08008000,   LENGTH = 352K
}
```

- It can be modified to create a bootloader for example
- If the origin is modified, make sure the offset in `system_stm32fxxx.c` file is also changed.
- Always refer to reference manual, datasheet and notes to have all pieces of information needed.

NEXT



- We will show you how to create a bootloader in the flash memory.
- Divide flash memory between the bootloader and application.
- Jump from bootloader to application.