NEOCRUX SYSTEMS

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)

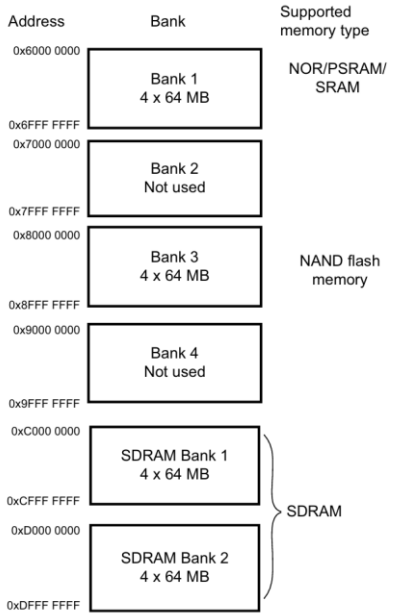
Flash Memory

- 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

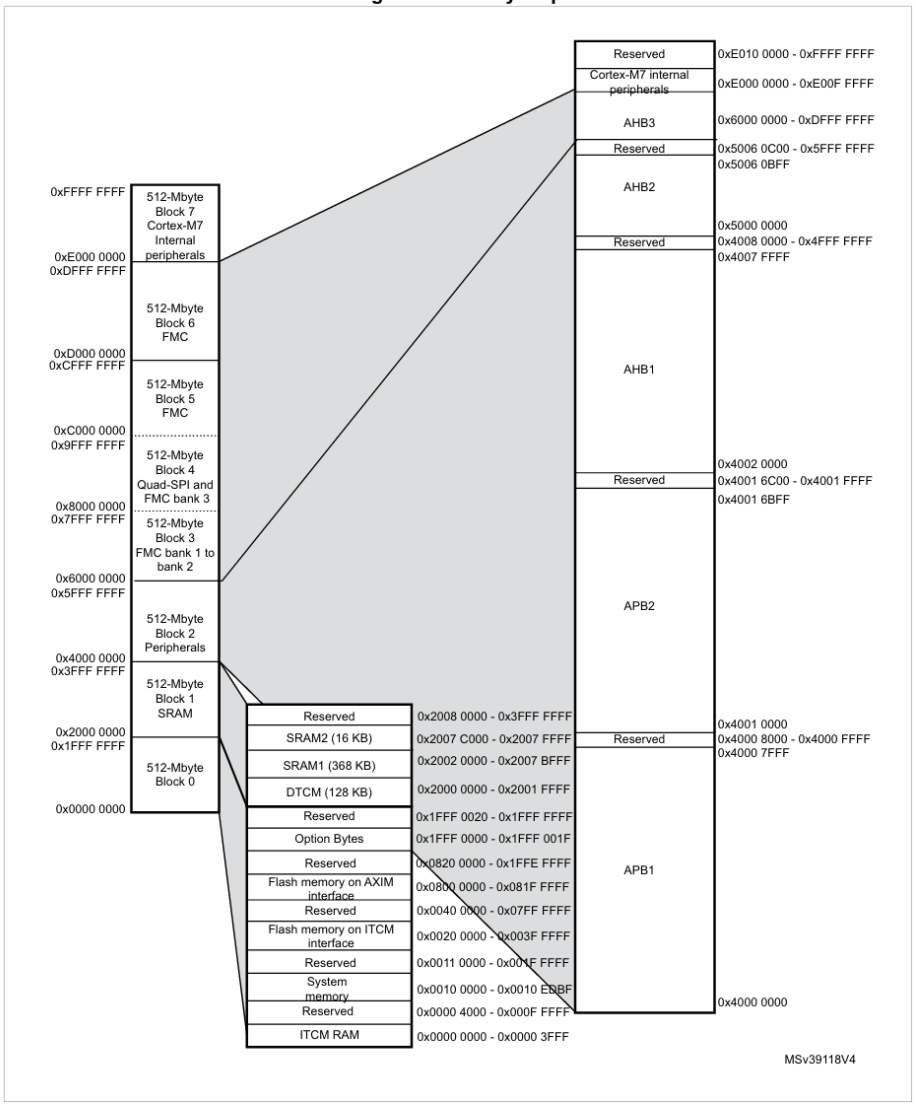
Registers

- 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

Use case: **STM32F767ZI**

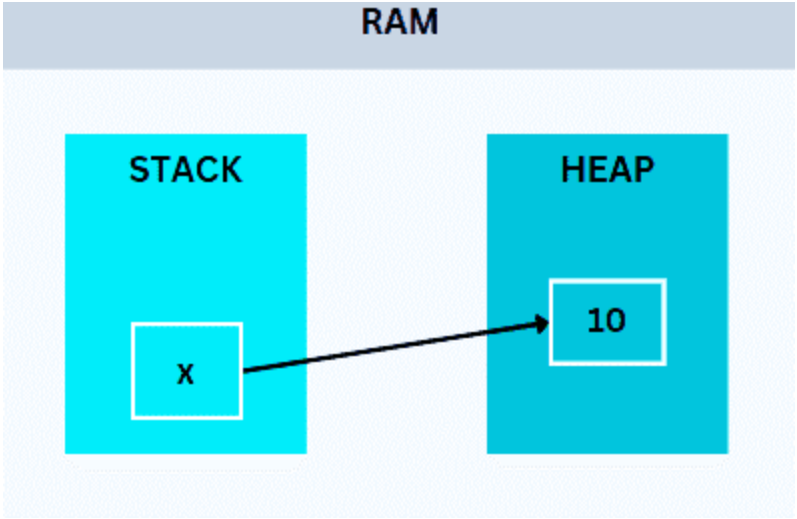
2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



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.

- 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.