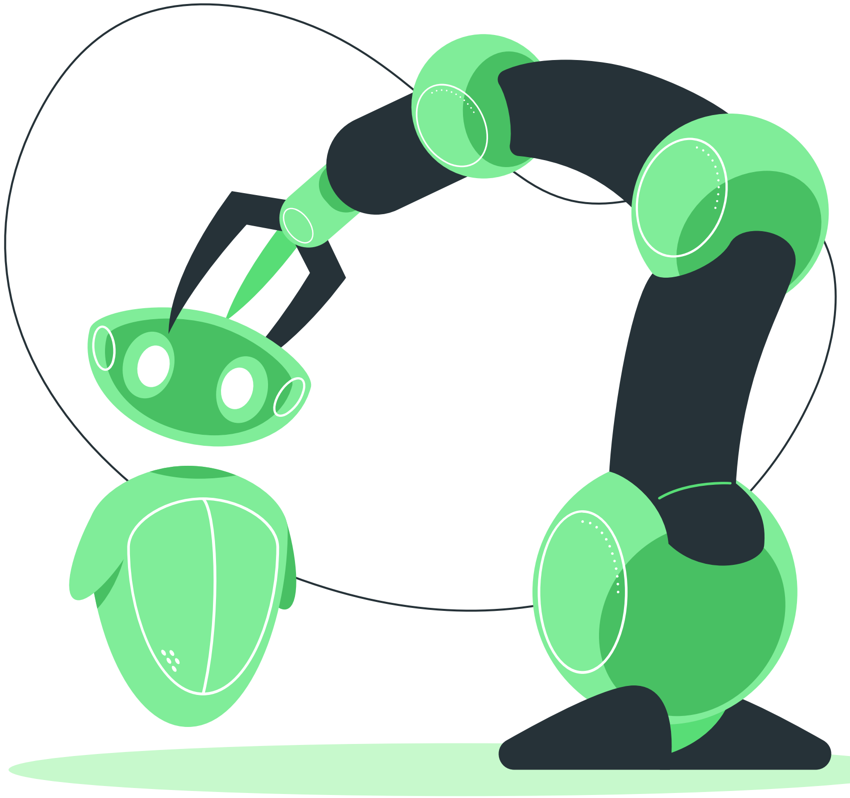
Freeways
free software club

STM32 Workshop

Session 3

By Moktar SELLAMI



Plan

1 ARM Processors

2 Memory layout

3 Interrupts and NVIC

4 Bus Matrix AMBA

5 RCC and Clock tree

6 Peripheral Configuration flow

ARM Processors

ARM (Advanced RISC Machines)

It is a RISC instruction set processor.

Known for **Low costs**, **low power** consumption, and **low heat** generation.

ARM Processors

ARM provides Multiple Processors:

A-Profile

CORTEX-X - NEOVERSE - CORTEX-A

ISA: ARMV8-A ARMV9-A

Complexe Apps:

TVs, Smartphones, Automotive Head units, Cloud storage.

ARM-R

CORTEX-R

ISA: ARMV8-R

Realtime & safety critical Apps:

ADAS, Vehicle Steering systems, Networking, Storage equipments.

ARM-M

CORTEX-M

ISA: ARMV8-M

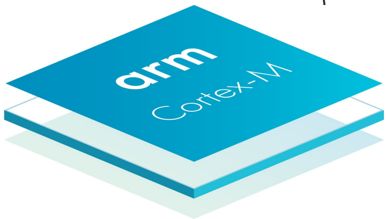
Mainstream Apps:

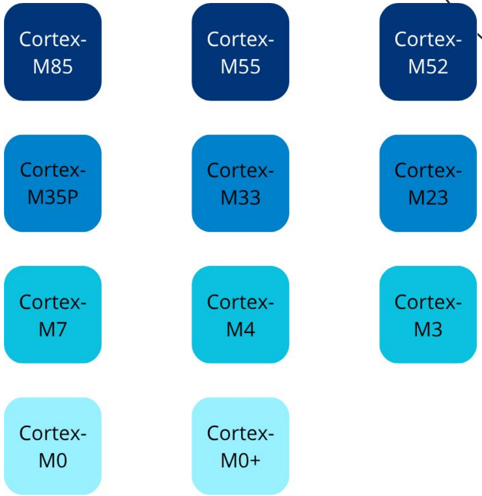
IoT, Embedded systems, wearables, Industry, smart homes...

ARM CORTEX-M Processors

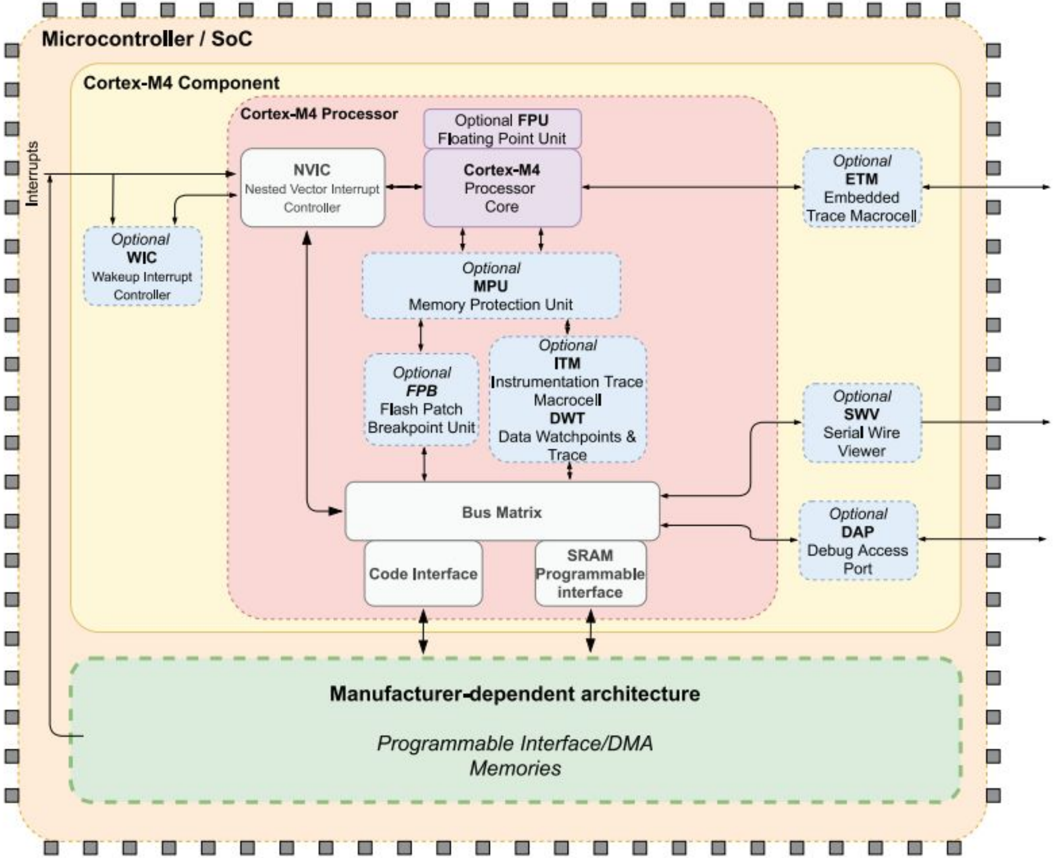
The ARM CORTEX-M Family

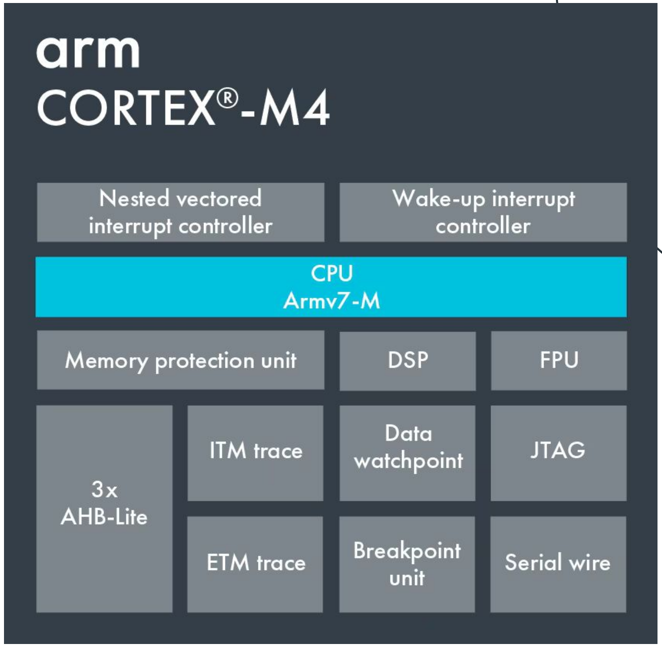
Cortex-M0	2009	Entry-level, ultra-low power	ARMv6 Simplest, smallest core, Low cost
Cortex-M0+	2012	Entry-level, ultra-low power	ARMv6
Cortex-M3	2006	Mainstream	Full ARMv7-M feature set,
Cortex-M4	2010	Mainstream + DSP	ARMv7E-M Adds DSP instructions and optional FPU
Cortex-M7	2014	High-performance	ARMv7 Dual-issue pipeline, higher clock speeds, FPU
Cortex-M23	2016	Secure entry-level	ARMv8-M baseline with TrustZone support
Cortex-M33	2016	Secure mainstream	ARMv8-M mainline + TrustZone + DSP/FPU options
Cortex-M55	2020	AI/ML focused	ARMv8.1-M + Helium Technology
Cortex-M85	2022	High-performance AI/ML	Helium Technology




Cortex-M85
Cortex-M55
Cortex-M52
Cortex-M35P
Cortex-M33
Cortex-M23
Cortex-M7
Cortex-M4
Cortex-M3
Cortex-M0
Cortex-M0+

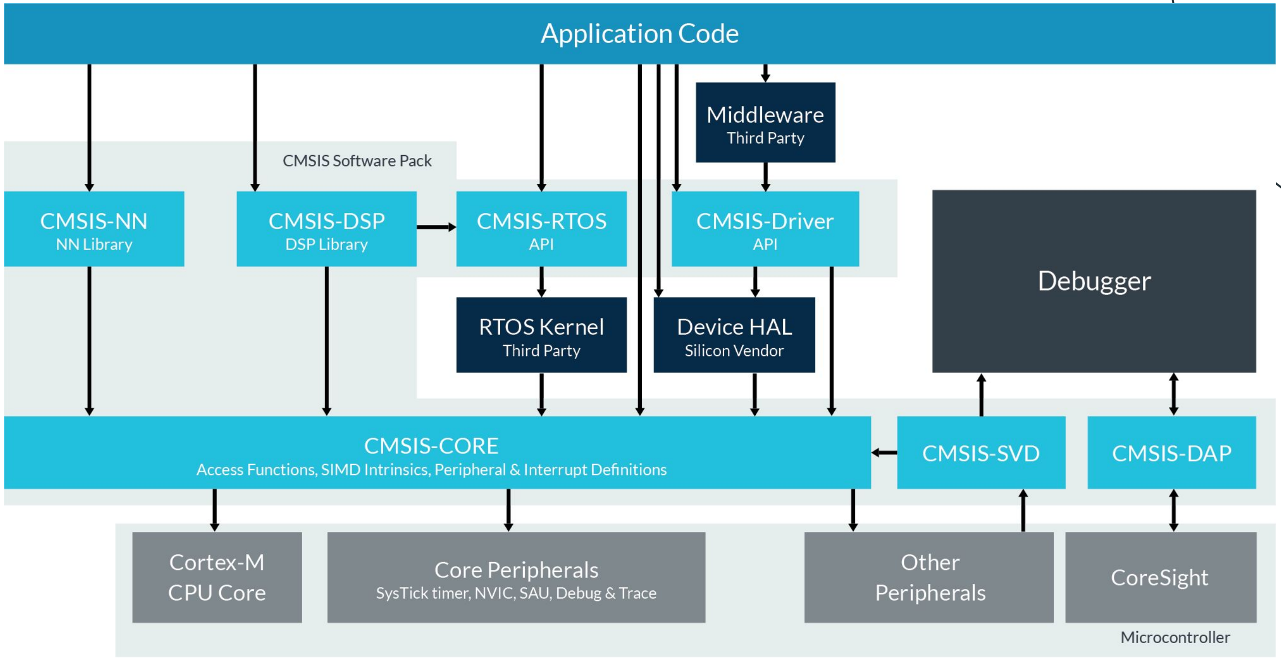
ARM CORTEX M–4





ARM CORTEX M-4: CMSIS

CMSIS (Cortex Microcontroller Software Interface Standard) is a vendor-independent **software** standard and a collection of **libraries** provided by ARM, designed to simplify software development on Cortex-M microcontrollers.



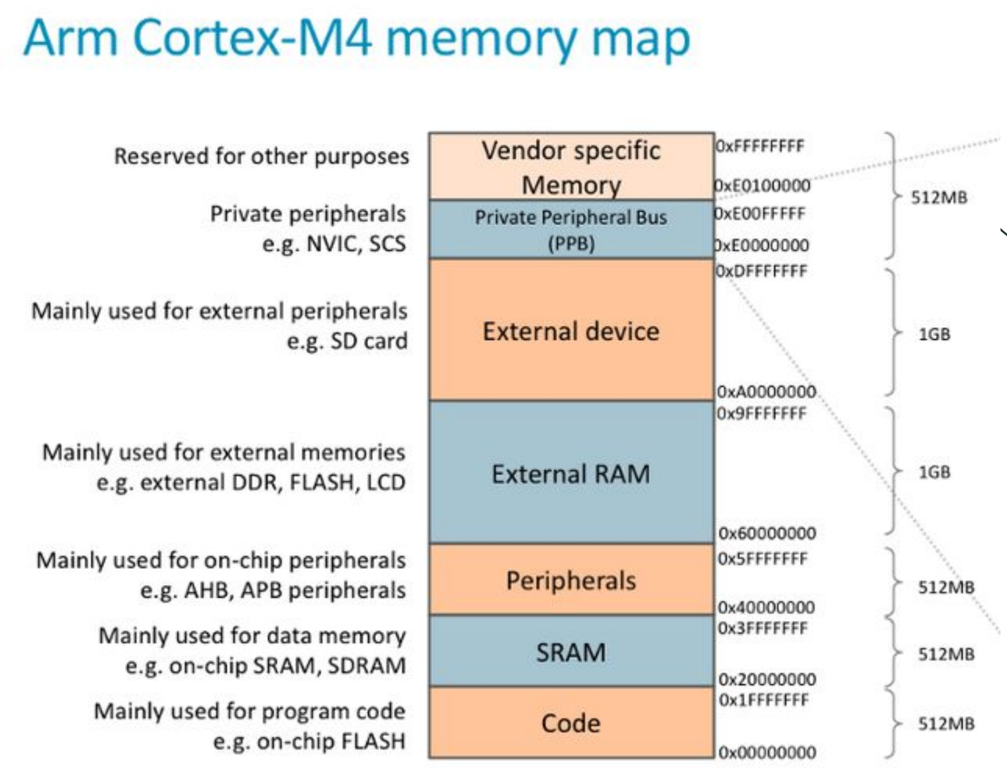
ARM CORTEX M–4: Thumb-2 ISA

ARM Thumb-2 ISA allows the intermixing of 32-bit instructions with the older 16-bit Thumb instructions. This facilitates maximum compatibility when running programs for the older architecture on the newer one.

💡 **Thumb-2** gives you almost the same performance as full 32-bit ARM instructions, but with much smaller code size.

Memory Map: Cortex M4

The Cortex-M4 processor implements the ARMv7-M architecture with a 32-bit address space and 32-bit data path.
32-bit address space, there are 2^32 memory positions: 4 (Gb) of memory space,

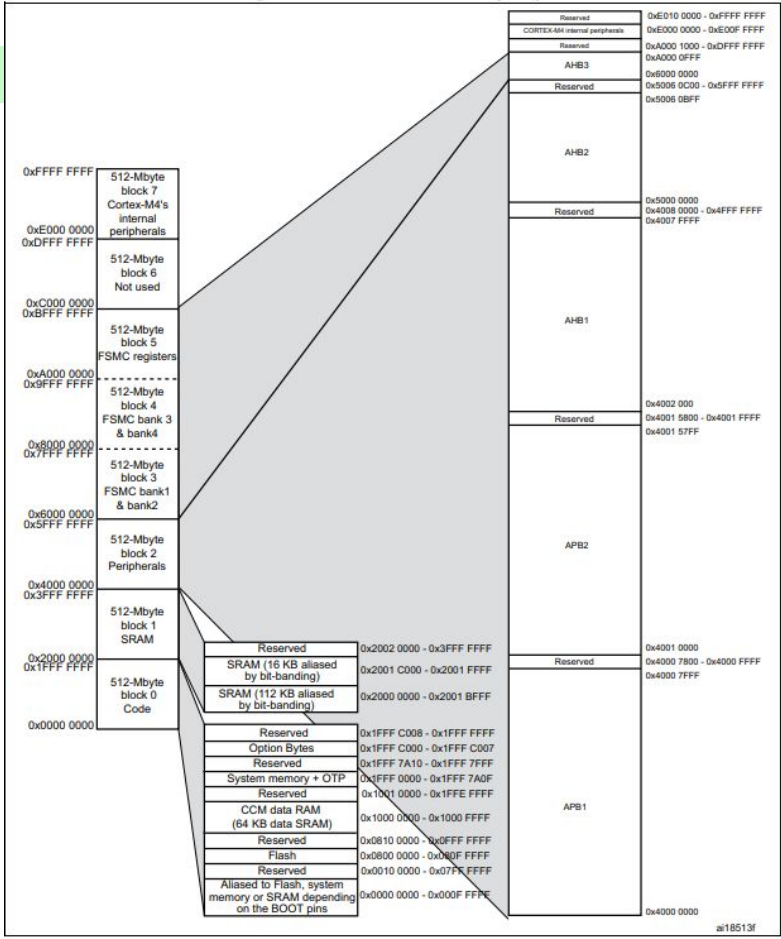


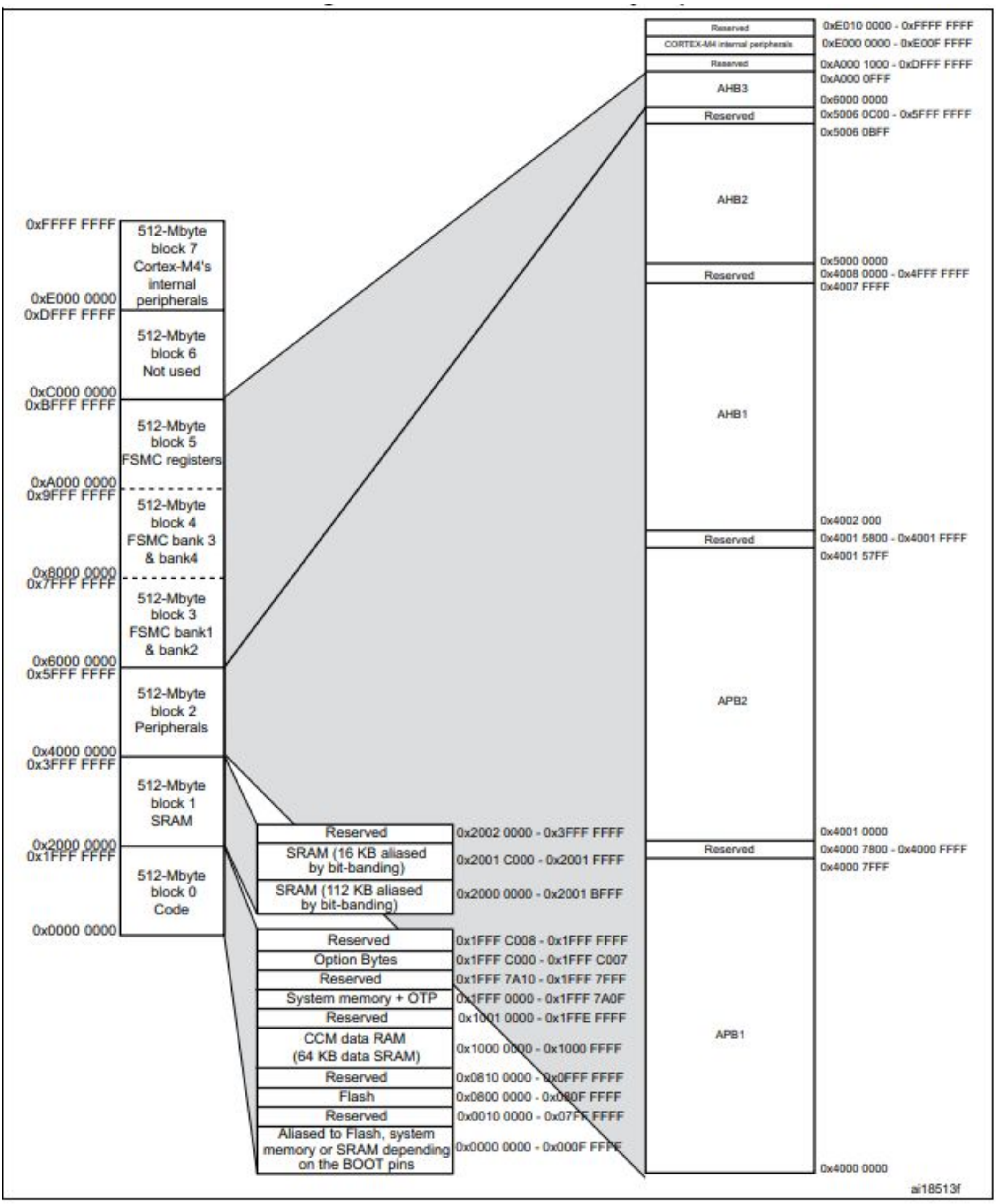
Arm Cortex-M4 memory map

Reserved for other purposes	Vendor specific Memory	0xFFFFFFFF 0xE0100000	512MB
Private peripherals e.g. NVIC, SCS	Private Peripheral Bus (PPB)	0xE00FFFFF 0xE0000000	
Mainly used for external peripherals e.g. SD card	External device	0xDFFFFFFF 0xA0000000	1GB
Mainly used for external memories e.g. external DDR, FLASH, LCD	External RAM	0x9FFFFFFF 0x60000000	1GB
Mainly used for on-chip peripherals e.g. AHB, APB peripherals	Peripherals	0x5FFFFFFF 0x40000000	512MB
Mainly used for data memory e.g. on-chip SRAM, SDRAM	SRAM	0x3FFFFFFF 0x20000000	512MB
Mainly used for program code e.g. on-chip FLASH	Code	0x1FFFFFFF 0x00000000	512MB

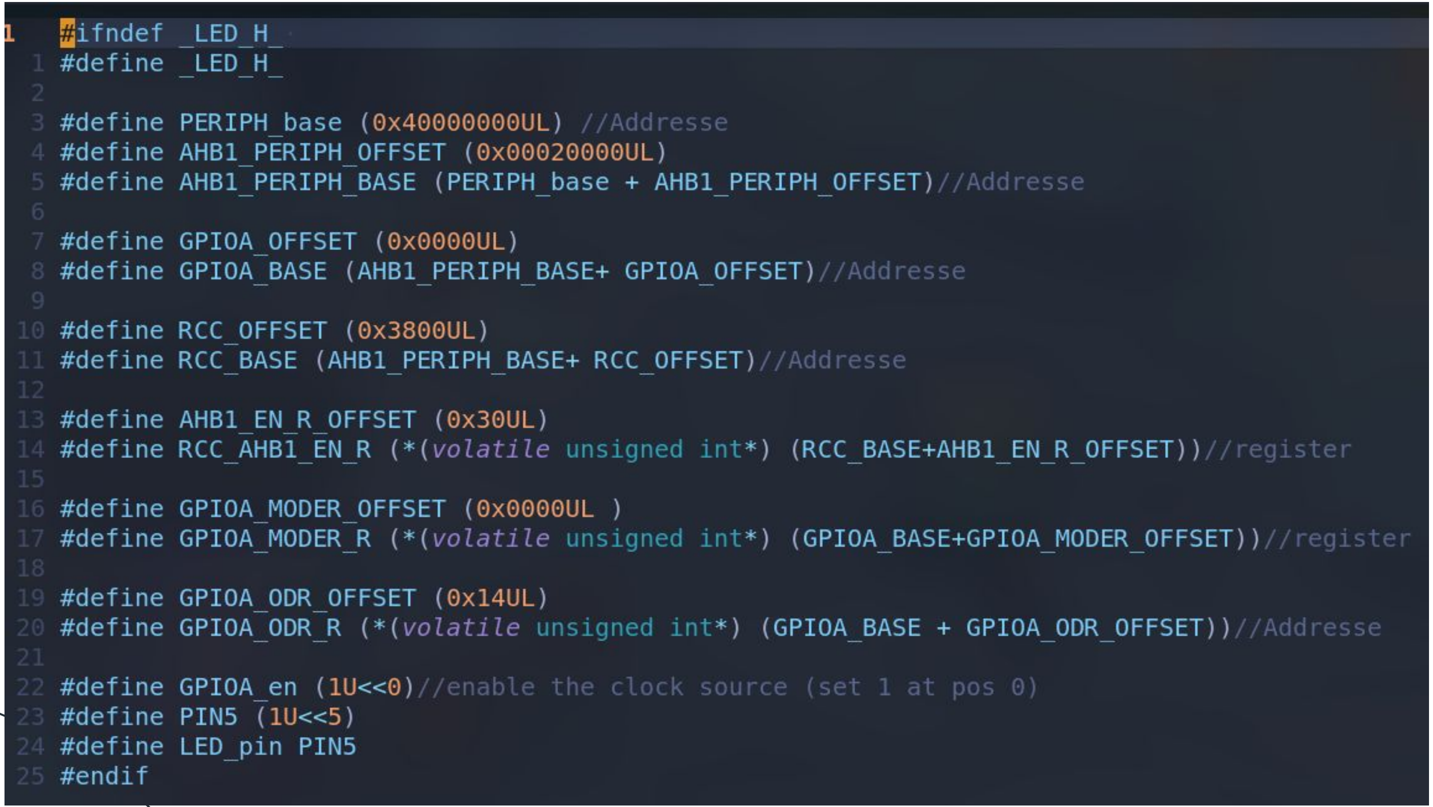
Memory Mapping STM32F407

Region	Start Address	End Address	Size
Code (Flash)	0x0000 0000	0x1FFF FFFF	512 MB
SRAM	0x2000 0000	0x3FFF FFFF	512 MB
Peripheral	0x4000 0000	0x5FFF FFFF	512 MB
External RAM	0x6000 0000	0x9FFF FFFF	1 GB
System	0xE000 0000	0xFFFF FFFF	512 MB


0xFFFF FFFF
512-Mbyte block 7 Cortex-M4's internal peripherals
0xE000 0000
0xDFFF FFFF
512-Mbyte block 6 Not used
0xC000 0000
0xBFFF FFFF
512-Mbyte block 5 FSMC registers
0xA000 0000
0x9FFF FFFF
512-Mbyte block 4 FSMC bank 3 & bank4
0x8000 0000
0x7FFF FFFF
512-Mbyte block 3 FSMC bank1 & bank2
0x6000 0000
0x5FFF FFFF
512-Mbyte block 2 Peripherals
0x4000 0000
0x3FFF FFFF
512-Mbyte block 1 SRAM
0x2000 0000
0x1FFF FFFF
512-Mbyte block 0 Code
0x0000 0000
Reserved
0x2002 0000 - 0x3FFF FFFF
SRAM (16 KB aliased by bit-banding)
0x2001 C000 - 0x2001 FFFF
SRAM (112 KB aliased by bit-banding)
0x2000 0000 - 0x2001 BFFF
Reserved
0x1FFF C008 - 0x1FFF FFFF
Option Bytes
0x1FFF C000 - 0x1FFF C007
Reserved
0x1FFF 7A10 - 0x1FFF 7FFF
System memory + OTP
0x1FFF 0000 - 0x1FFF 7A0F
Reserved
0x1001 0000 - 0x1FFE FFFF
CCM data RAM (64 KB data SRAM)
0x1000 0000 - 0x1000 FFFF
Reserved
0x0810 0000 - 0x0FFF FFFF
Flash
0x0800 0000 - 0x080F FFFF
Reserved
0x0010 0000 - 0x07FF FFFF
Aliased to Flash, system memory or SRAM depending on the BOOT pins
0x0000 0000 - 0x000F FFFF
Reserved
0xE010 0000 - 0xFFFF FFFF
CORTEX-M4 internal peripherals
0xE000 0000 - 0xE00F FFFF
Reserved
0xA000 1000 - 0xDFFF FFFF
AHB3
0xA000 0FFF
0x6000 0000
Reserved
0x5006 0C00 - 0x5FFF FFFF
0x5006 0BFF
AHB2
0x5000 0000
Reserved
0x4008 0000 - 0x4FFF FFFF
0x4007 FFFF
AHB1
0x4002 000
Reserved
0x4001 5800 - 0x4001 FFFF
0x4001 57FF
APB2
0x4001 0000
Reserved
0x4000 7800 - 0x4000 FFFF
0x4000 7FFF
APB1
0x4000 0000
ai18513f

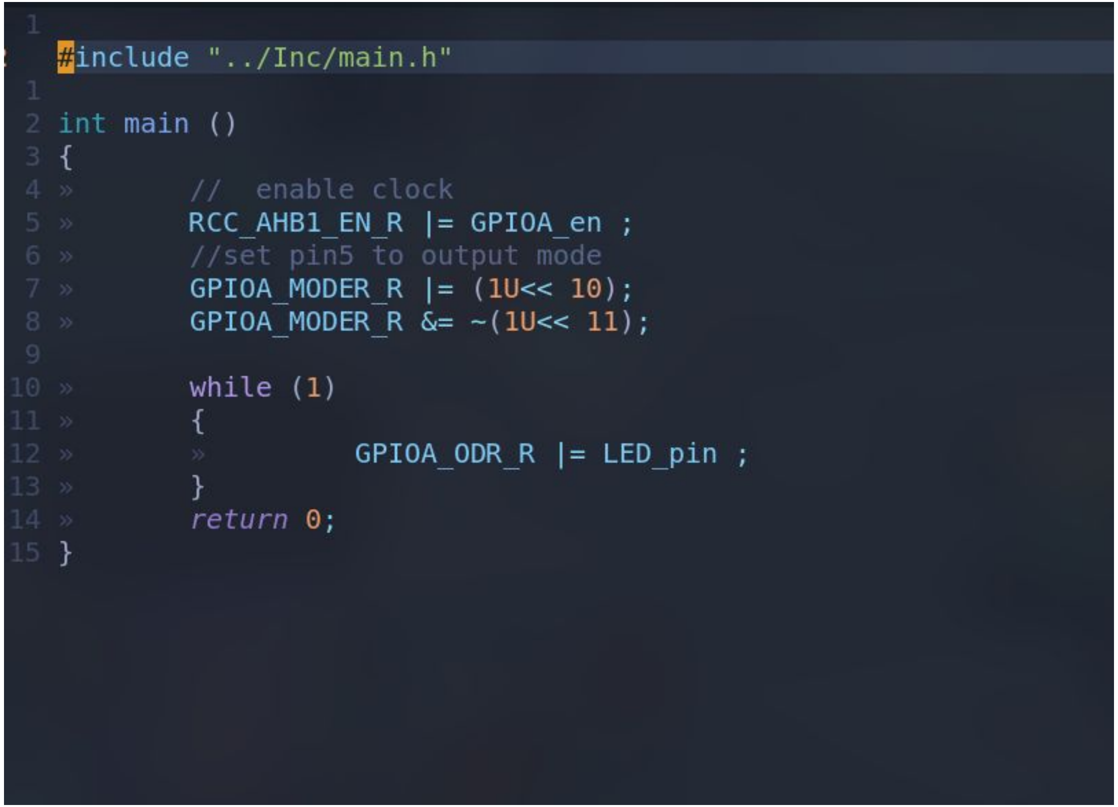

0xFFFF FFFF
512-Mbyte block 7 Cortex-M4's internal peripherals
0xE000 0000
0xDFFF FFFF
512-Mbyte block 6 Not used
0xC000 0000
0xBFFF FFFF
512-Mbyte block 5 FSMC registers
0xA000 0000
0x9FFF FFFF
512-Mbyte block 4 FSMC bank 3 & bank4
0x8000 0000
0x7FFF FFFF
512-Mbyte block 3 FSMC bank1 & bank2
0x6000 0000
0x5FFF FFFF
512-Mbyte block 2 Peripherals
0x4000 0000
0x3FFF FFFF
512-Mbyte block 1 SRAM
0x2000 0000
0x1FFF FFFF
512-Mbyte block 0 Code
0x0000 0000
Reserved
0x2002 0000 - 0x3FFF FFFF
SRAM (16 KB aliased by bit-banding)
0x2001 C000 - 0x2001 FFFF
SRAM (112 KB aliased by bit-banding)
0x2000 0000 - 0x2001 BFFF
Reserved
0x1FFF C008 - 0x1FFF FFFF
Option Bytes
0x1FFF C000 - 0x1FFF C007
Reserved
0x1FFF 7A10 - 0x1FFF 7FFF
System memory + OTP
0x1FFF 0000 - 0x1FFF 7A0F
Reserved
0x1001 0000 - 0x1FFE FFFF
CCM data RAM (64 KB data SRAM)
0x1000 0000 - 0x1000 FFFF
Reserved
0x0810 0000 - 0x0FFF FFFF
Flash
0x0800 0000 - 0x080F FFFF
Reserved
0x0010 0000 - 0x07FF FFFF
Aliased to Flash, system memory or SRAM depending on the BOOT pins
0x0000 0000 - 0x000F FFFF
Reserved
0xE010 0000 - 0xFFFF FFFF
CORTEX-M4 internal peripherals
0xE000 0000 - 0xE00F FFFF
Reserved
0xA000 1000 - 0xDFFF FFFF
AHB3
0xA000 0FFF
0x6000 0000
Reserved
0x5006 0C00 - 0x5FFF FFFF
0x5006 0BFF
AHB2
0x5000 0000
Reserved
0x4008 0000 - 0x4FFF FFFF
0x4007 FFFF
AHB1
0x4002 000
Reserved
0x4001 5800 - 0x4001 FFFF
0x4001 57FF
APB2
0x4001 0000
Reserved
0x4000 7800 - 0x4000 FFFF
0x4000 7FFF
APB1
0x4000 0000
ai18513f

Memory Mapping: STM32F407



```
#ifndef _LED_H_
#define _LED_H_

#define PERIPH_base (0x40000000UL) //Addresse
#define AHB1_PERIPH_OFFSET (0x00020000UL)
#define AHB1_PERIPH_BASE (PERIPH_base + AHB1_PERIPH_OFFSET)//Addresse

#define GPIOA_OFFSET (0x0000UL)
#define GPIOA_BASE (AHB1_PERIPH_BASE+ GPIOA_OFFSET)//Addresse

#define RCC_OFFSET (0x3800UL)
#define RCC_BASE (AHB1_PERIPH_BASE+ RCC_OFFSET)//Addresse

#define AHB1_EN_R_OFFSET (0x30UL)
#define RCC_AHB1_EN_R (*(volatile unsigned int*) (RCC_BASE+AHB1_EN_R_OFFSET))//register

#define GPIOA_MODER_OFFSET (0x0000UL )
#define GPIOA_MODER_R (*(volatile unsigned int*) (GPIOA_BASE+GPIOA_MODER_OFFSET))//register

#define GPIOA_ODR_OFFSET (0x14UL)
#define GPIOA_ODR_R (*(volatile unsigned int*) (GPIOA_BASE + GPIOA_ODR_OFFSET))//Addresse

#define GPIOA_en (1U<<0)//enable the clock source (set 1 at pos 0)
#define PIN5 (1U<<5)
#define LED_pin PIN5
#endif
```

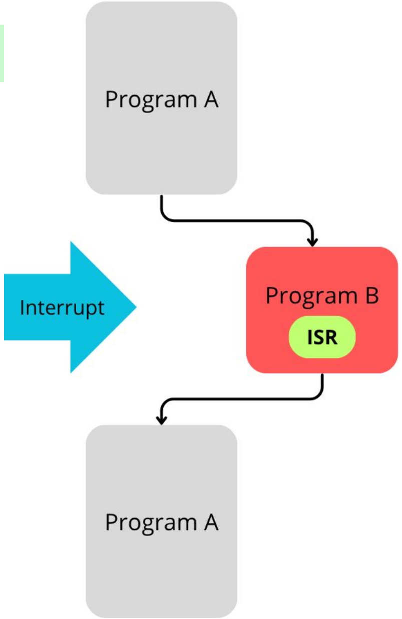
Memory Mapping: STM32F407



```c
#include "../Inc/main.h"

int main ()
{
        //  enable clock
        RCC_AHB1_EN_R |= GPIOA_en ;
        //set pin5 to output mode
        GPIOA_MODER_R |= (1U<< 10);
        GPIOA_MODER_R &= ~(1U<< 11);

        while (1)
        {
                GPIOA_ODR_R |= LED_pin ;
        }
        return 0;
}
```

Interrupts:

An Interrupt is a signal that tells the CPU to pause its current program, execute a specific function called an Interrupt Service Routine (ISR) to handle the event, and then seamlessly return to what it was doing.

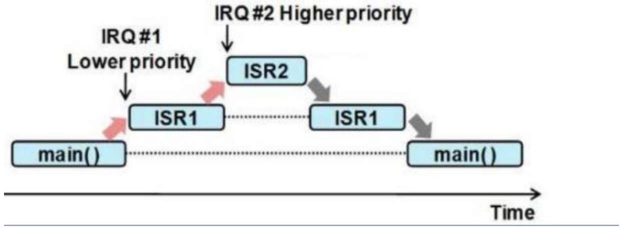


Execution Mode of a CPU:

- Pooling Mode
- Interrupt Mode
- DMA

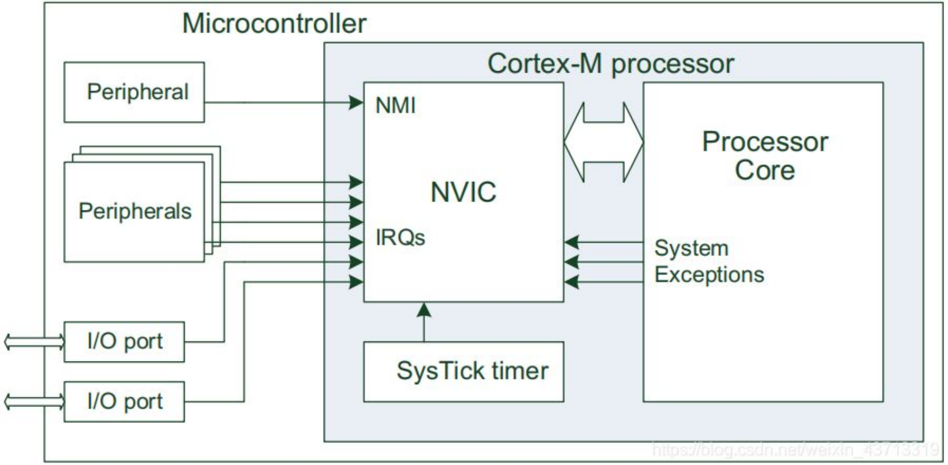
Interrupt source

- Externel
- Internal



Interrupts: ARM Cortex M4 NVIC

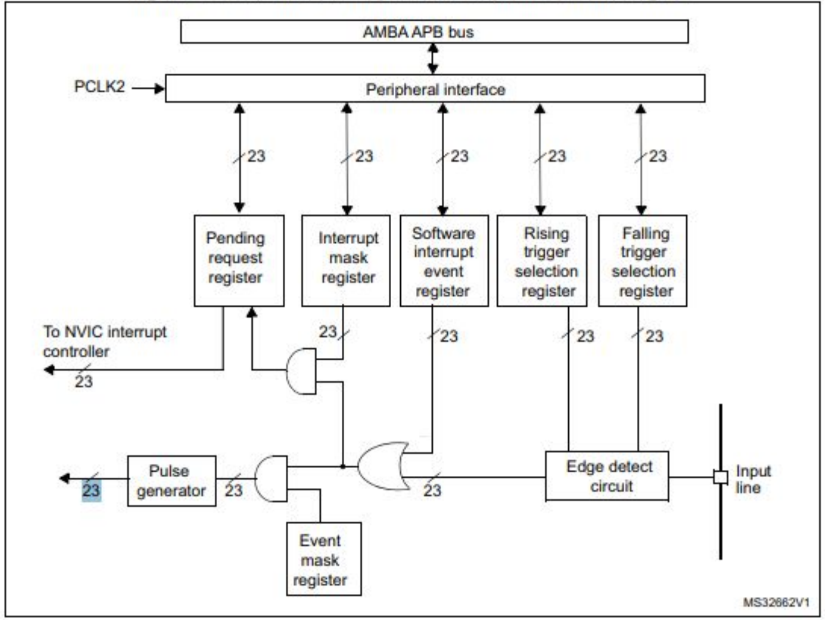
NVIC: The Nested Vectored Interrupt Controller is the dedicated hardware unit inside an Arm Cortex-M processor that manages all these interrupts efficiently. It's the dispatcher that handles the fire alarms.



External Interrupts: NVIC STM32F407

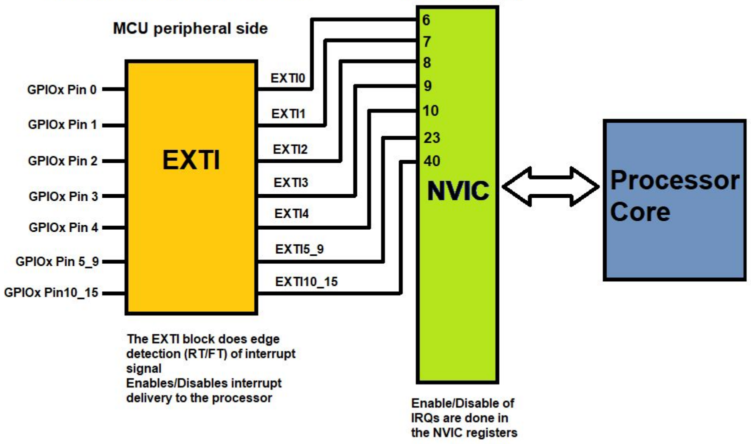
The **EXTI** (External Interrupt/Event Controller) is a peripheral in the STM32F407 that specializes in detecting external pin changes and generating interrupts or events.

Figure 41. External interrupt/event controller block diagram



External Interrupts: STM32F407 EXTI

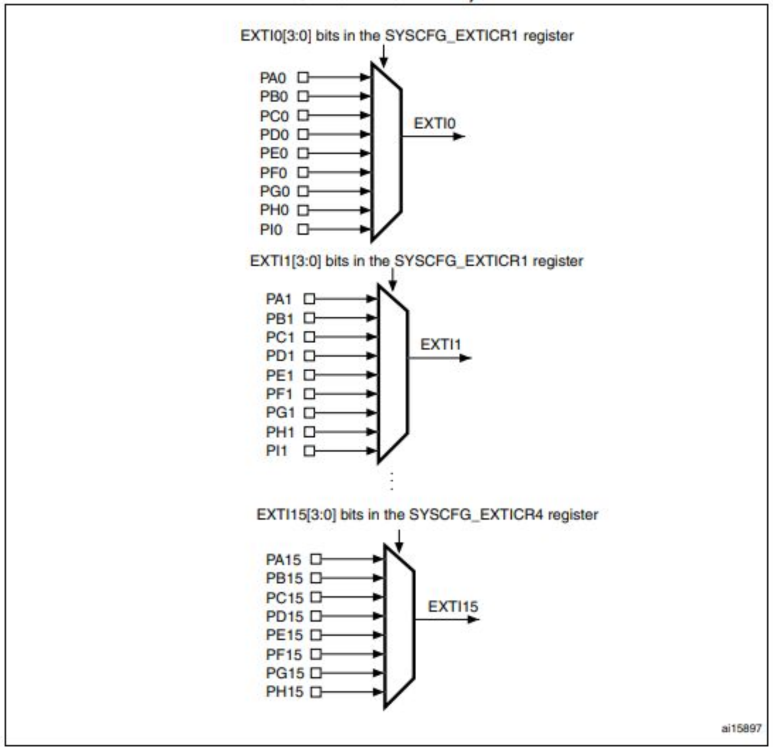
STM32F407xx Microcontroller GPIO Pins
Interrupt Delivery to the Processor



The EXTI block does edge detection (RT/FT) of interrupt signal
Enables/Disables interrupt delivery to the processor

Enable/Disable of IRQs are done in the NVIC registers

STM32F415xx/17xx)


EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
PH0
PI0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
PH1
PI1
EXTI1
EXTI15[3:0] bits in the SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
PH15
EXTI15
ai15897

Internal Interrupts: CORTEX M4

Exceptions in Cortex-M: Internal interrupts

Reset: Runs after system reset.

NMI (Non-Maskable Interrupt): Highest priority interrupt, cannot be disabled; used for critical events (e.g., power-down).

HardFault: Generic fault for unhandled errors (instruction or system faults).

MemManage: Handles memory protection faults (via MPU).

BusFault: Handles bus access errors (instruction or data).

UsageFault: Handles invalid instruction or operation errors.

Bus Matrix: AMBA

Bus Matrix: AMBA

AHB: Advanced High performance Bus (186Mhz)

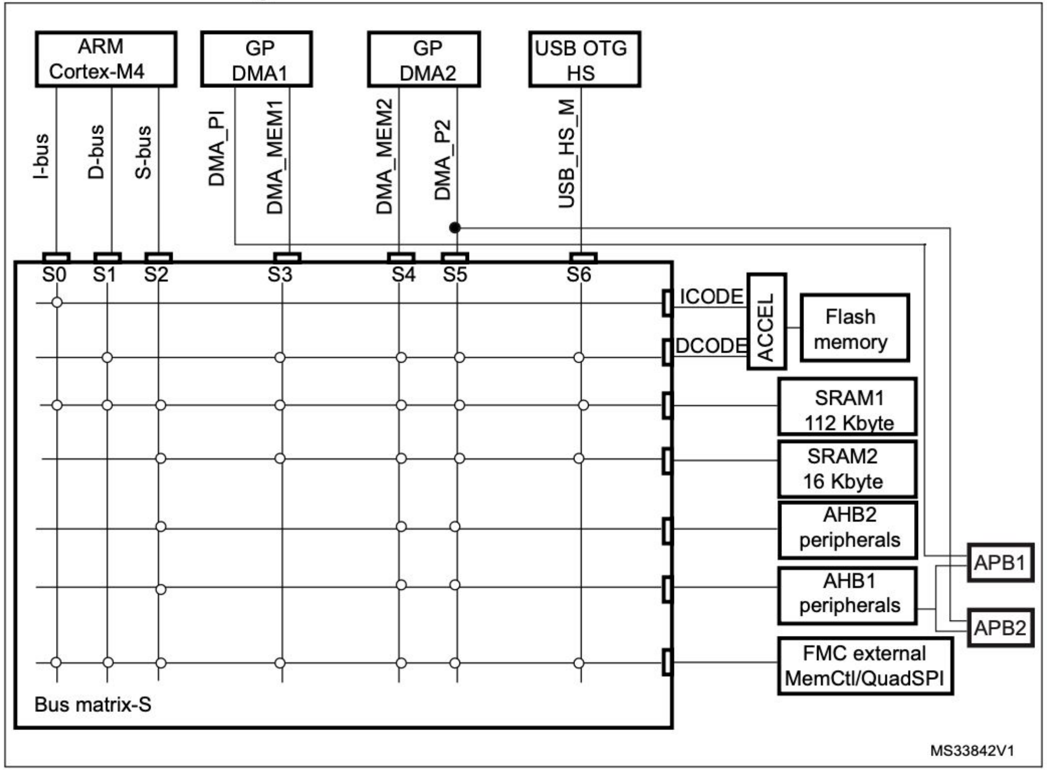
APB: Advanced peripheral Bus (84Mhz)

I-BUS

D-Bus

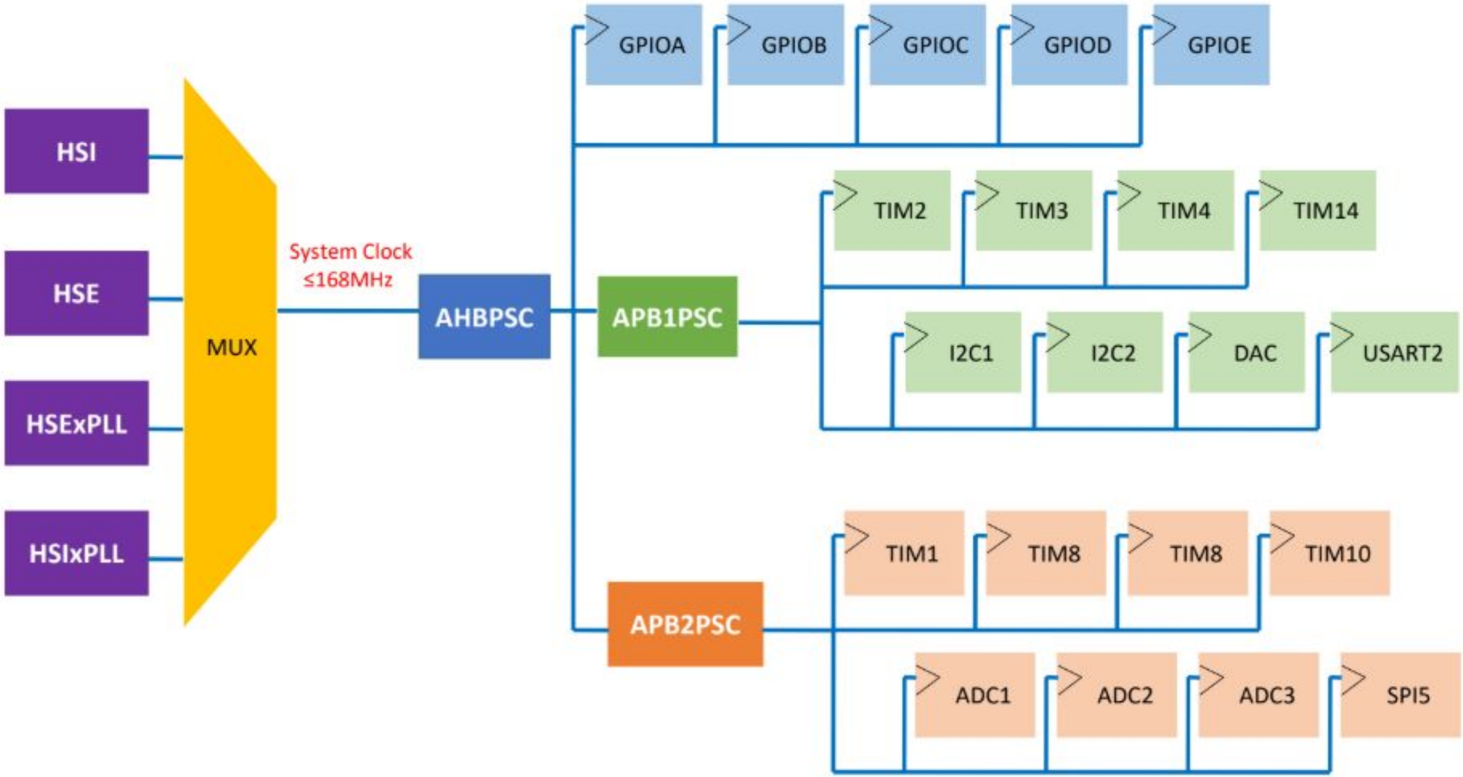
S-Bus

Figure 4. STM32F446xC/E and Multi-AHB matrix

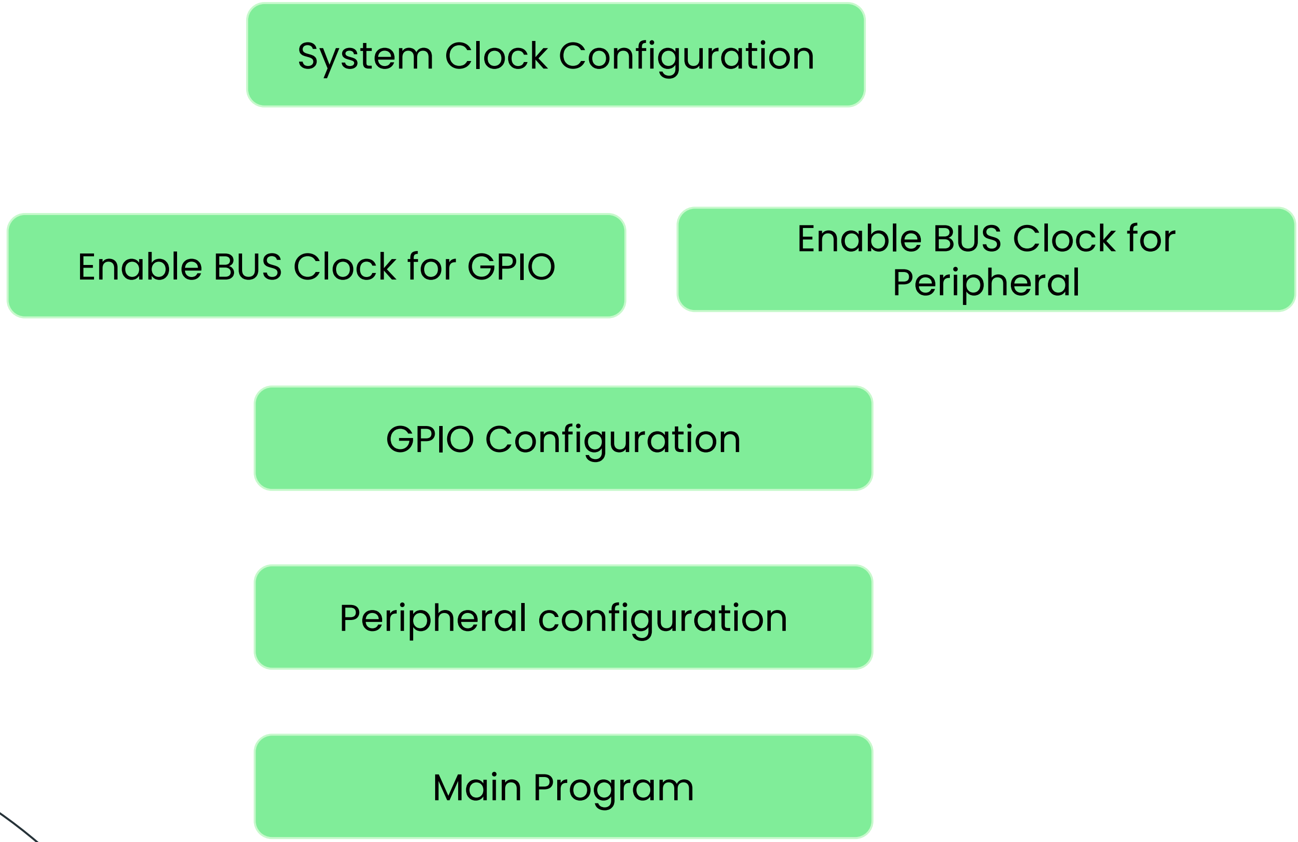


RCC and clock Tree

RCC and clock Tree



Peripheral Configuration flow - Generic



System Clock Configuration

Enable BUS Clock for GPIO

Enable BUS Clock for Peripheral

GPIO Configuration

Peripheral configuration

Main Program

Peripheral Configuration flow - Generic

System Clock Configuration: Set up the main system clock (HSI, HSE, PLL) and bus prescalers.

Enable BUS Clock for GPIO: Enable the clock for the GPIO port(s) used by the peripheral.

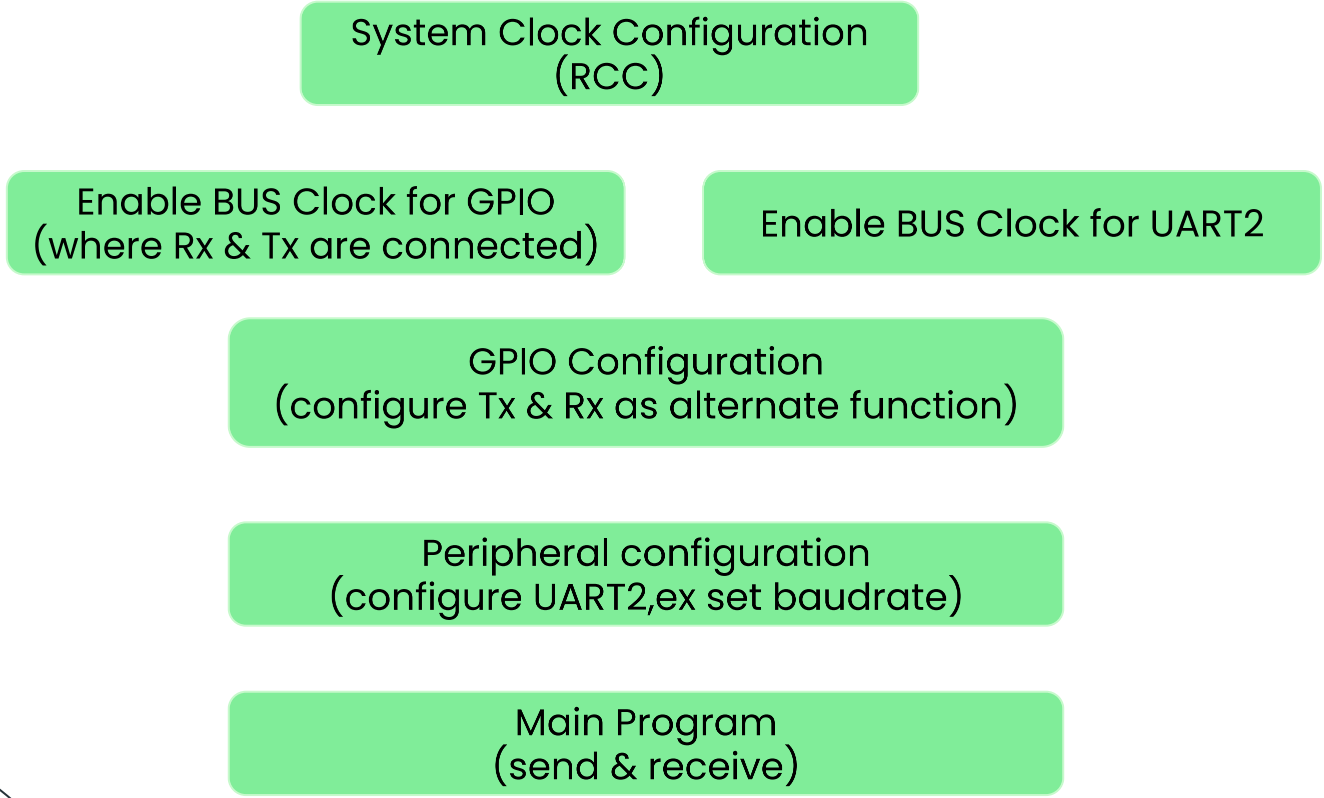
Enable BUS Clock for Peripheral: Enable the specific peripheral’s clock (e.g., USART, SPI, I2C).

GPIO Configuration: Configure GPIO pins (Mode, Speed, PullUP/PullDown)

Peripheral Configuration: Initialize and configure the peripheral itself .

Main Program: Your main loop or application logic using the configured peripheral.

Peripheral Configuration flow : Use case UART2



System Clock Configuration (RCC)

Enable BUS Clock for GPIO (where Rx & Tx are connected)

Enable BUS Clock for UART2

GPIO Configuration (configure Tx & Rx as alternate function)

Peripheral configuration (configure UART2,ex set baudrate)

Main Program (send & receive)

Freeways
free software club

Thank You