LABORATORIO DI ARCHITETTURE E PROGRAMMAZIONE DEI SISTEMI ELETTRONICI INDUSTRIALI

Laboratory Lesson 3:

- EXTernal Interrupt (EXTI)

- FLASH Memory

Prof. Luca Benini <luca.benini@unibo.it>

Filippo Casamassima <filippo.casamassima@unibo.it>
Domenico Balsamo <domenico.balsamo@unibo.it>

Course Organization

- Hands-on session LAB1 **Thursday 15.00 – 19.00**
- Prof Benini Friday **9.00 – 11.00 room 5.5**
- Lab is available **Friday 11.00 – 13.00**
- Check website for **announcements, course material:** http://www-micrel.deis.unibo.it/LABARCH
- Final Exam:
  - Homeworks (**to be checked weekly**)
  - Final project
  - Final discussion (homeworks + final project)

Peripheral Usage Recap

- How to use a Peripheral using standard Peripheral Library:
  - Enable Peripheral Clock
    - RCC_AHB1PeriphClockCmd(PeriphName, ENABLE);
    - Look in stm32f4xx_rcc.h for PeriphName (row 318)
  - Fill in Configuration Structure PeriphName_InitTypeDef
  - Call Init Function:
  - PeriphName_Init(&PeriphName_InitStructure);
  - Use the Peripheral ☺

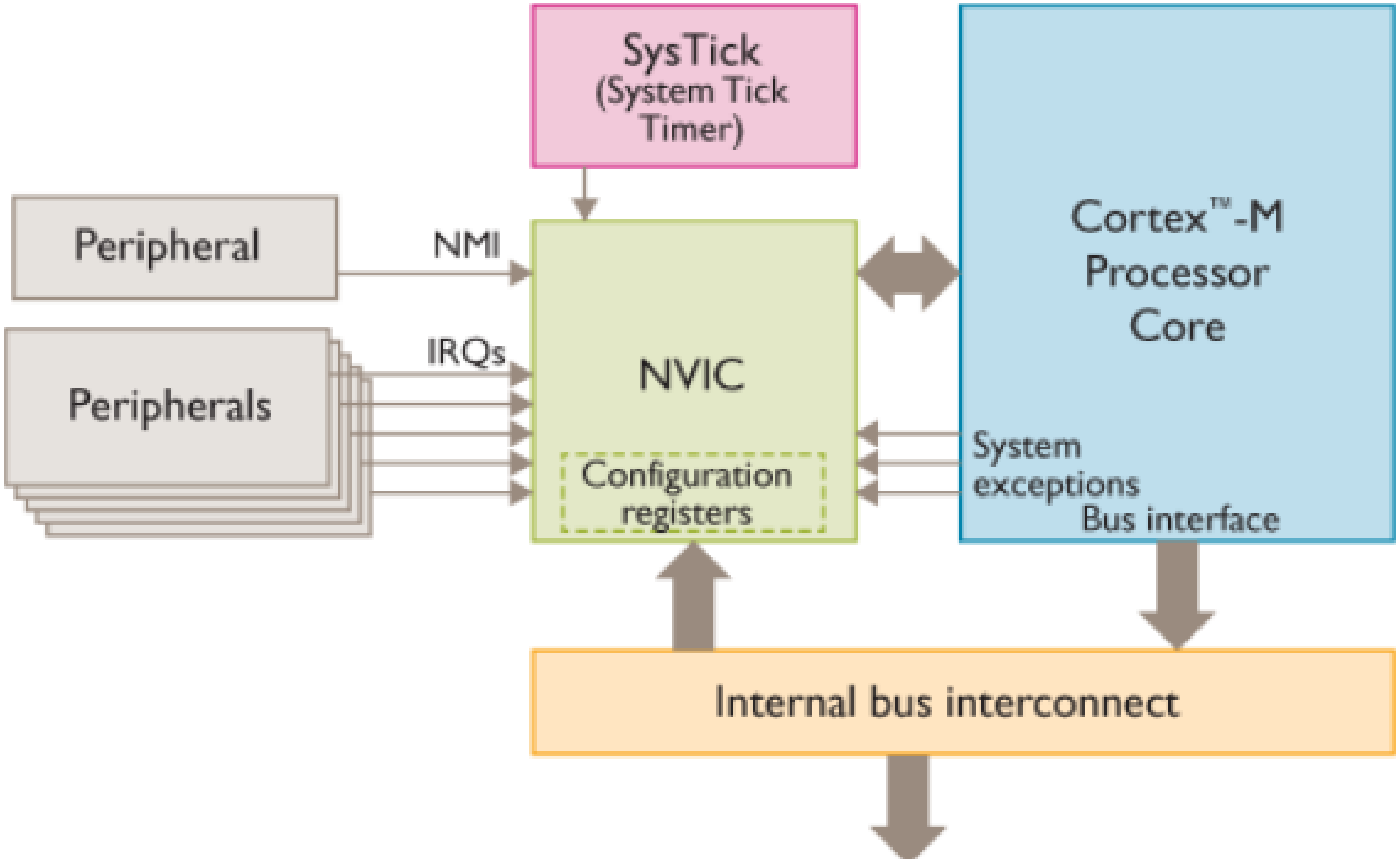
IDE Troubleshooting

- To debug project:
  - Right click project name ->Debug As… ->AC6 STM32 C/C++
- If you have errors that should not be present try:
  - Right Click Project Name -> Index ->Rebuild
- If you cannot start the debugger:
  - Check that the board is connected and you have RED or GREEN Led on then one of the following:
    - If you have red and Green led blinking, you have another debug session active, terminate it and restart
    - If Red and Green led are both on (orange light) there is a communication error, disconnect and reconnect the board
    - Go to Task Manager and terminate “openocd” process

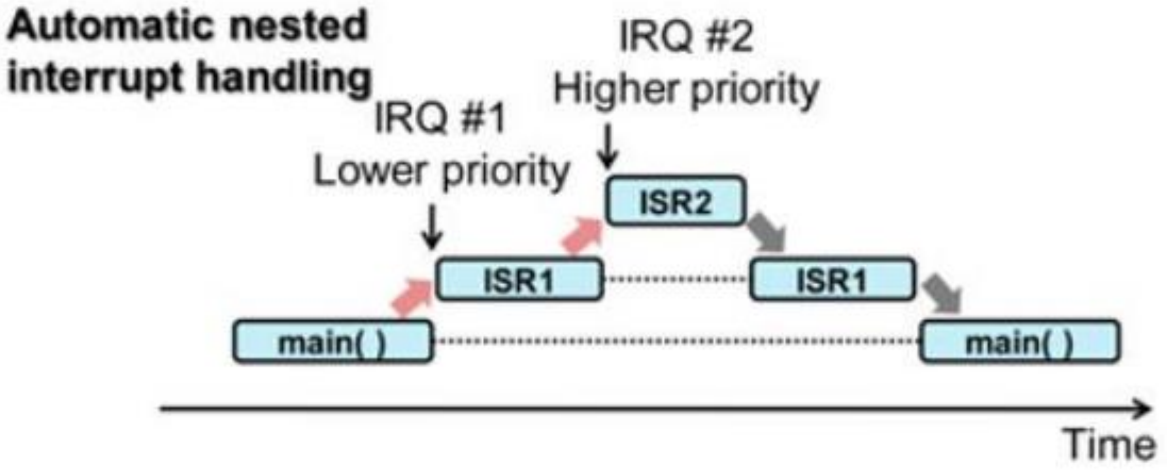
Nested Vector Interrupt Controller NVIC

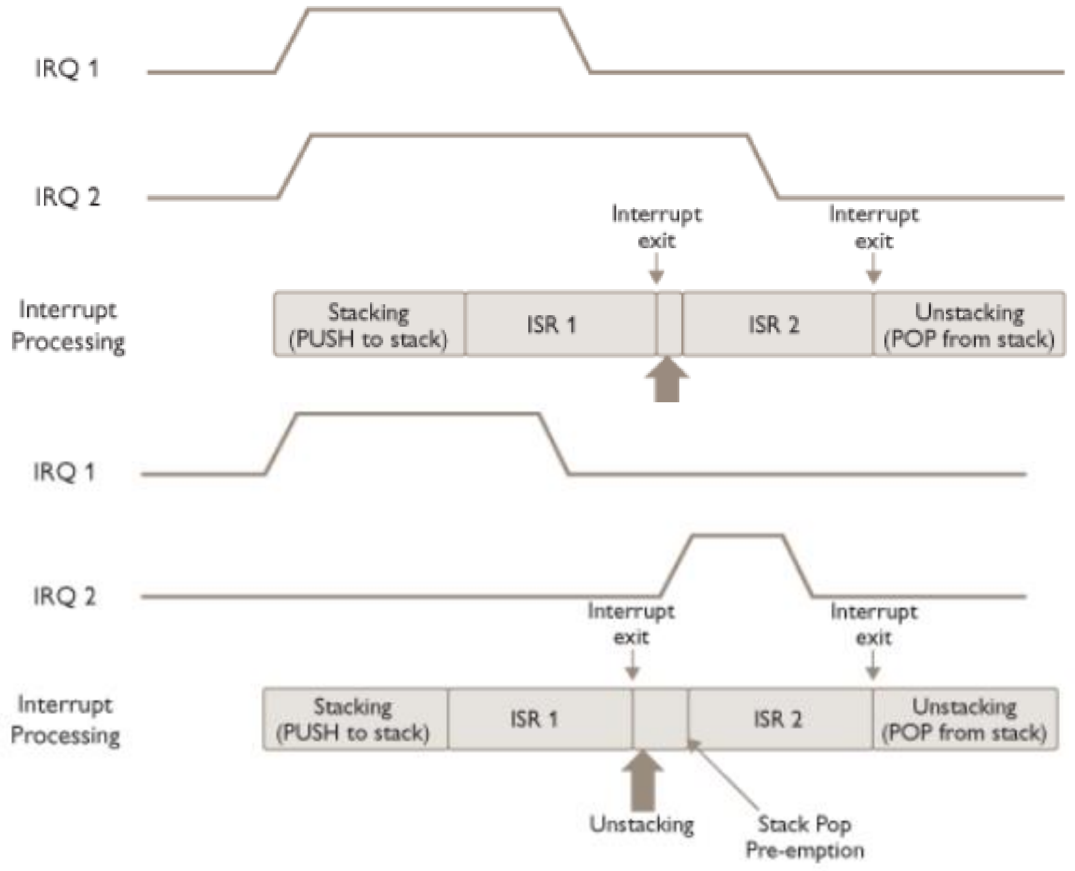
- Up to 81 interrupts (depends on the STM32 device type)
- Programmable priority level of 0-15
  - A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority
- Dynamic reprioritization of interrupts
- Grouping of priority values into group priority and sub-priority fields Interrupt tail-chaining
- An external Non-maskable interrupt (NMI)

Nested Vector Interrupt Controller NVIC



Nested Vector Interrupt Controller NVIC





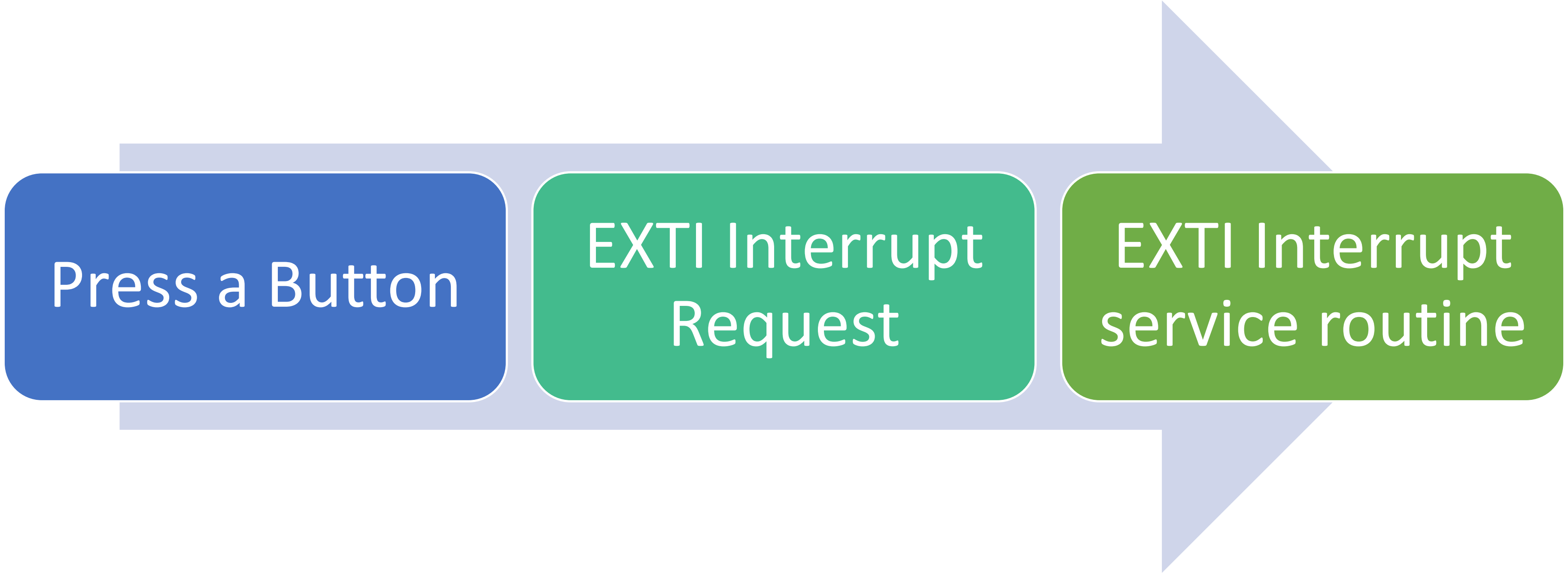
- Nested Interrupt: If a interrupt request (IRQ) with higher priority is raised, it is served first

- Tail chaining: for nested ISR does not restore all saved registers from the stack.
- Stack pop pre-emption: If another exception occurs during the unstacking process of an exception, the processor abandons the stack Pop

3. EXTernal Interrupt
EXTI

EXTI – Purpose

- We want to configure an external interrupt line.
- An EXTI line is configured to generate an interrupt on each falling edge.
- In the interrupt routine a led connected to a specific GPIO pin is toggled.



Press a Button

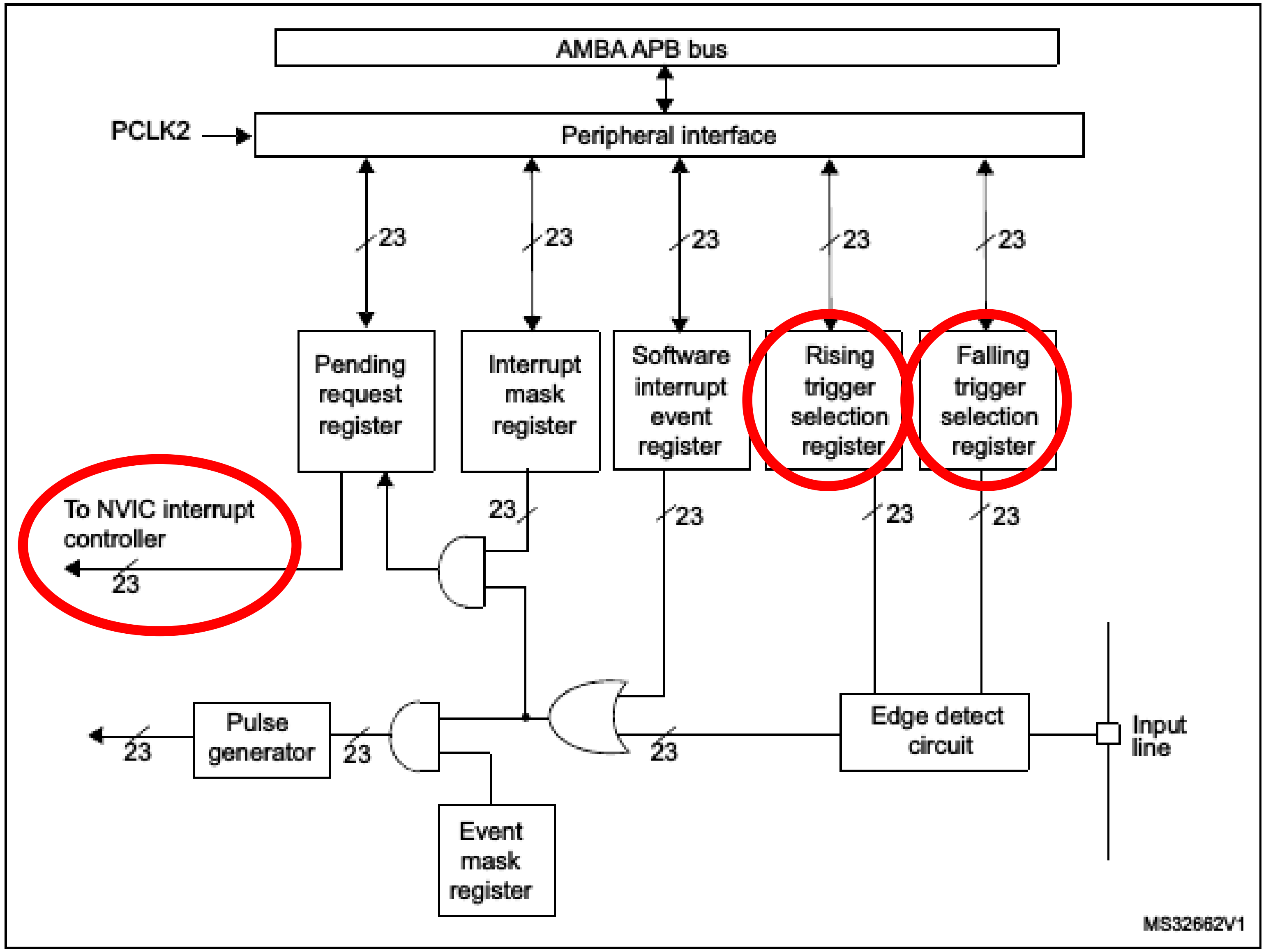
EXTI Interrupt Request

EXTI Interrupt service routine

EXTI - How

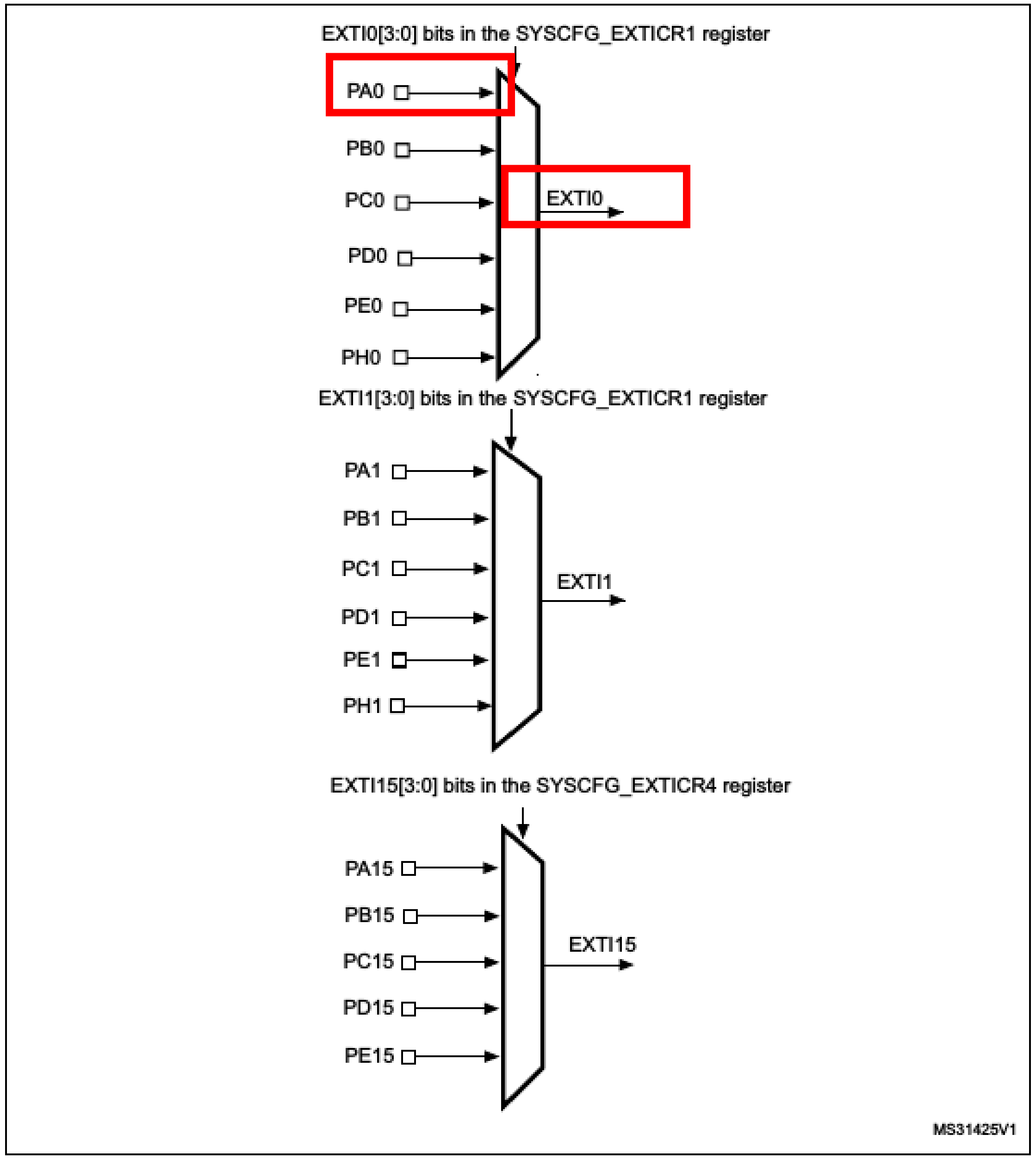
- Look at the Reference Manual: RM0368 Chapter 10

Figure 29. External interrupt/event controller block diagram



EXTI - Capabilities

Figure 30. External interrupt/event GPIO mapping


EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PH0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PH1
EXTI1
EXTI15[3:0] bits in the SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
EXTI15
MS31425V1

- There are 16 EXTI lines connected to GPIOs
- All pins with the same pin number are connected on the same EXTI line (eg. Pin_2 Port A and Pin_2 Port C share the same EXTI2)
- EXTI 16 – 22 are reserved for RTC, USB etc…

EXTI – Turn on led, on Button pressed

I want to turn on a LED using the button. **What do I need to know? (UM1669)**

Which bus LEDs and the button are connected to?

- GPIO ports are always on the **AHB1** bus

Which port are we going to use for LEDS and the button?

- Leds are on port GPIOD, pins 12 - 15

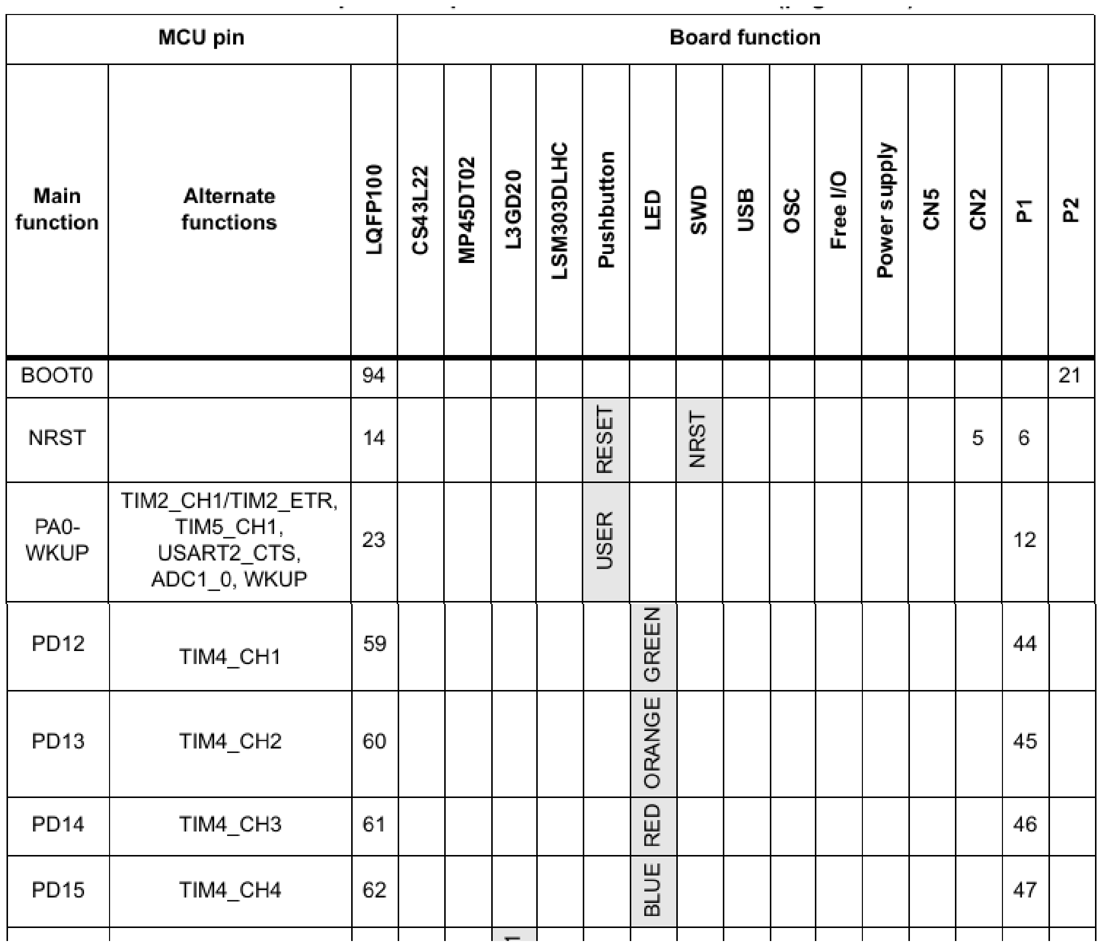
Which port and Pin are we going to use for the Button ?

- Pushbutton is on port GPIOA pin 0

What do I need to do with this GPIO? (input, output, ...)

I need to write (output) for the LEDs

I need to read (input) for the button



MCU pin			Board function														
Main function	Alternate functions	LQFP100	CS43L22	MP45DT02	L3GD20	LSM303DLHC	Pushbutton	LED	SWD	USB	OSC	Free I/O	Power supply	CN5	CN2	P1	P2
BOOT0		94															21
NRST		14					RESET		NRST						5	6	
PA0-WKUP	TIM2_CH1/TIM2_ETR, TIM5_CH1, USART2_CTS, ADC1_0, WKUP	23					USER									12	
PD12	TIM4_CH1	59						GREEN								44	
PD13	TIM4_CH2	60						ORANGE								45	
PD14	TIM4_CH3	61						RED								46	
PD15	TIM4_CH4	62						BLUE								47	

EXTI – CODE

```
/*LED INITIALIZATION*/
void LedSetup(void){
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable the GPIO_LED Clock */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

    /* Configure the GPIO_LED pin */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

EXTI – CODE

```c
/* INITALIZE BUTTON AND INTERRUPT */
void ButtonSetup(void){
    GPIO_InitTypeDef GPIO_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;
    EXTI_InitTypeDef EXTI_InitStructure;

    /* Enable GPIOA clock */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

    /* Configure PA0 pin as input floating */
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```

EXTI – CODE

```
/* Enable SYSCFG clock */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

/* Configure EXTI Line0 */
EXTI_InitStructure.EXTI_Line = EXTI_Line0;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

/* Connect EXTI Line0 to PA0 pin */
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);

/* Enable and set EXTI Line0 Interrupt to the lowest priority */
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

EXTI – CODE

```
/**
  * @brief This function handles External line 0 interrupt 
request.
  * @param None
  * @retval None
  */
void EXTI0_IRQHandler(void)
{
  if(EXTI_GetITStatus(EXTI_Line0) != RESET)
  {
    /* Toggle LED4 */
   STM_EVAL_LEDToggle(LED4);

    /* Clear the EXTI line 0 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line0);
  }
}
```

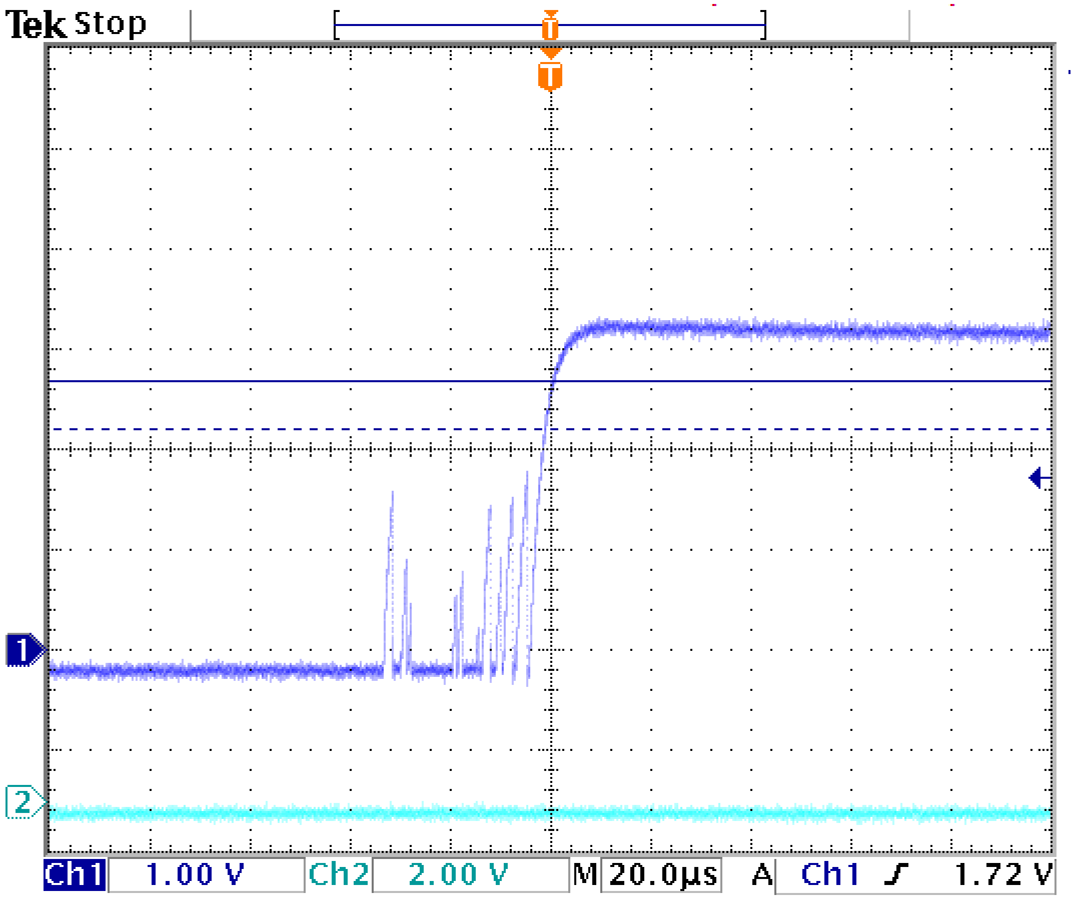
What Happens?

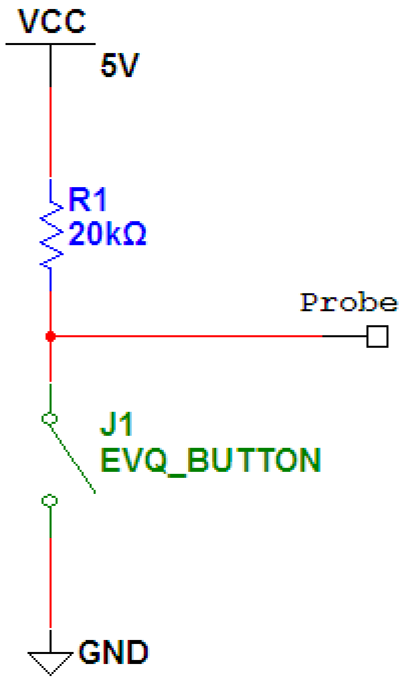
- The Microcontroller is always in the for loop, when an interrupt is detected the ISR void `EXTI0_IRQHandler(void){}` is called

```
int main(void)
{
	LedSetup();
	ButtonSetup();
	for(;;);
}
```

EXTI (Debounce)

Contact switches have tiny little springs on them that pop them up when you release the button. This is great for popping the button, but terrible as a digital input. This is because the springs, when releases, bounces around and creates instability for a short duration.





This duration of the instability is on the order of several milliseconds, usually no more than 10ms. In human terms, this is not such a big deal, but a microcontroller processing at 24 MHz would read the input as multiple button presses instead of just one.

EXTI – Questions

- Guarda il manual della scheda (UM1669)
- 3.a Nella struttura di Inizializzazione settiamo `GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING` per il bottone perchè non servono Pull-up/down?
- 3.b Cosa succede se impostiamo `EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;` ?
- 3.c Quali registri (guarda sul manual RM0368) sono utilizzati per far scattare l'interrupt sul fronte di salita/discesa?
- 3.d Quale parametron della strttura `EXTI_InitStructure.` va usato per avere un trigger su entrambi I fronti (salita/discesa) ?

EXTI - Esercizi

- 3.1 Scrivi un codice per evitare il debounce del bottone
- 3.2 Scrivi un codice in cui premendo il bottone viene fatto partire il lampeggio del LED
- 3.3 Scrivi un codice che riconosce il doppio click e fa variare la frequenza di lampeggio del led
- *3.4 Scrivi un codice che riconosca l'interrupt sul pin PC6 quando questo viene connesso a GND. (Suggerimento: configura la linea con pull-up e cerca il nome della routine nel file di startup) (facoltativo)*

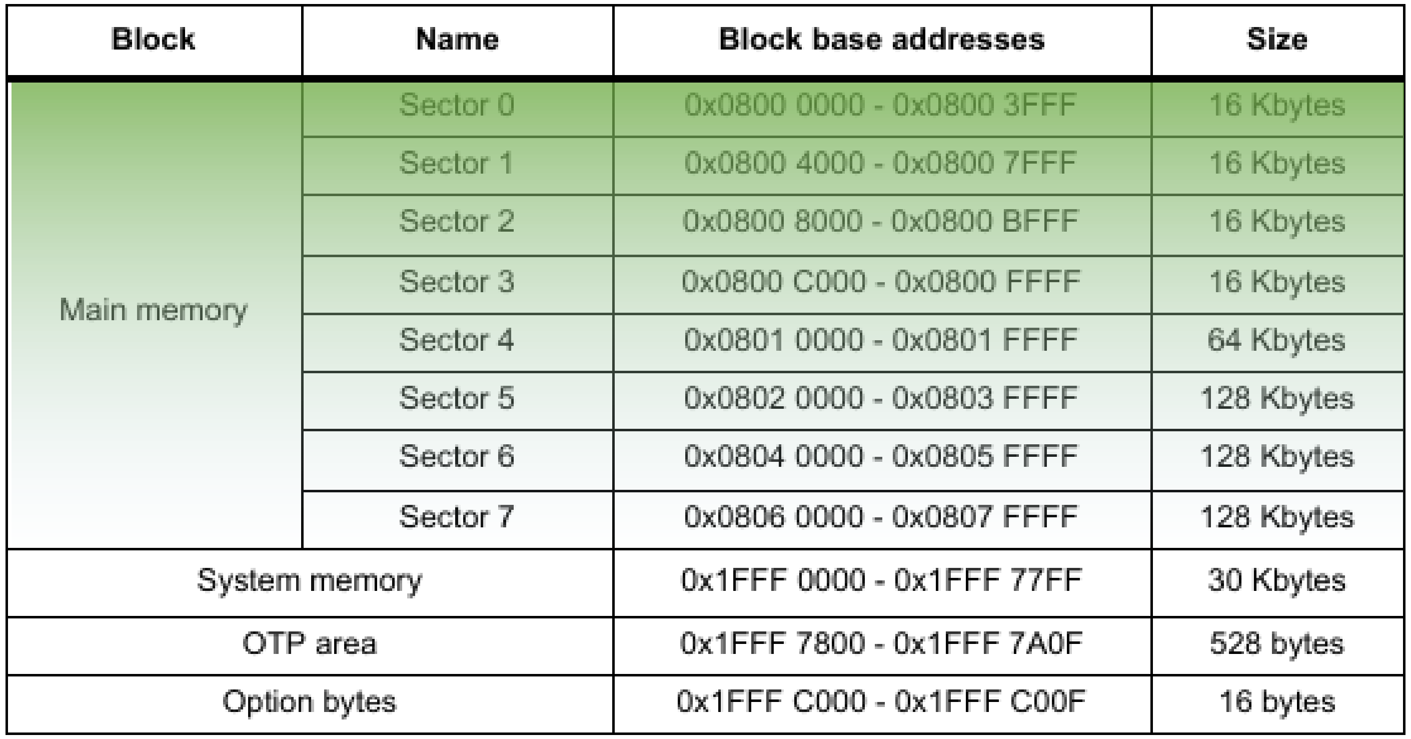
4. FLASH Memory

FLASH – Characteristics

- Flash memory is a non-volatile computer storage chip that can be electrically erased and reprogrammed
- Flash memory is non-volatile, meaning no power is needed to maintain the information stored in the chip
- Flash memory offers fast read access times (although not as fast as volatile DRAM memory used for main memory in PCs) and better kinetic shock resistance than hard disks
- Limitations: - although it can be read or programmed a byte or a word at a time in a random access fashion, it can only be erased a "block" at a time (once a bit has been set to 0, only by erasing the entire block can it be changed back to 1) - another limitation is that flash memory has a finite number of program-erase cycles (typically in the order of 10K)

FLASH – Memory Organization

Table 5. Flash module organization (STM32F401xB/C and STM32F401xD/E)



Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbytes
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
	Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
System memory		0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP area		0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes		0x1FFF C000 - 0x1FFF C00F	16 bytes

Flash is Memory Mapped: can be accesses trough its address

Your code is written from address 0x0800 0000

Your Board has 256Kbyte of FLASH, how many blocks are available?

FLASH – How To Write

- Look at the Manual RM0368 Chapter 3.5
- Look at file stm32f4xx_flash.h for available functions
- FLASH memory is write protected after programming
- Erase occurs one sector at a time
  - WARNING: you could erase sectors with your own code
- Programming occur one word at a time, BYSY flag must be checked
- After programming, lock memory to avoid undesired writing

FLASH – Key functions

- I want to store data in flash. How can I store data in flash memory?

1. Unlock the Flash Bank1
   - Using void FLASH_UnlockBank1(void);
2. Clear All pending flags
   - Using void FLASH_ClearFlag(uint32_t FLASH_FLAG);
3. Erase the FLASH sectors
   - Using FLASH_Status FLASH_EraseSector(i, VoltageRange_3) ;
4. Program Flash Bank1
   - Using FLASH_Status FLASH_ProgramWord(uint32_t Address, uint32_t Data);
5. Lock the Flash Bank1
   - Using void FLASH_LockBank1(void);

FLASH - Code

-
  ```
  /* Unlock the Flash to enable the flash control register access
  *************/
    FLASH_Unlock();

    /* Erase the user Flash area
    (area defined by FLASH_USER_START_ADDR and FLASH_USER_END_ADDR)
  ***********/

    /* Clear pending flags (if any) */
    FLASH_ClearFlag(FLASH_FLAG_EOP | FLASH_FLAG_OPERR | FLASH_FLAG_WRPERR |
                    FLASH_FLAG_PGAERR | FLASH_FLAG_PGPERR|FLASH_FLAG_PGSERR);



      /* Call EraseSectorFunction to clean the sector */
      if (FLASH_EraseSector(FLASH_Sector_3, VoltageRange_3) != FLASH_COMPLETE)
      {
        /* Error occurred while sector erase. LED ON */
        while (1)
        { STM_EVAL_LEDOn(LED5); }
      }
  ```
-

FLASH – Code

-
  ```
  /* Program the user Flash area word by word
    (area defined by FLASH_USER_START_ADDR and FLASH_USER_END_ADDR) ***********/

    Address = FLASH_USER_START_ADDR;

    while (Address < FLASH_USER_END_ADDR)
    {
      if (FLASH_ProgramWord(Address, DATA_32) == FLASH_COMPLETE)
      {
        Address = Address + 4;
      }
      else
      {
        /* Error occurred while writing data in Flash memory.LED ON */
        while (1)
        {STM_EVAL_LEDOn(LED5); }
      }
    }

    /* Lock the Flash to disable the flash control register access (recommended
    to protect the FLASH memory against possible unwanted operation) *********/
    FLASH_Lock();
  ```
-

FLASH – Code

```c
/* Check if the programmed data is OK*/
  Address = FLASH_USER_START_ADDR;
  MemoryErrors = 0x0;

  while (Address < FLASH_USER_END_ADDR)
  {
    data32 = *(uint32_t*)Address;

    if (data32 != DATA_32)
    {
      MemoryErrors++;
      /* Data is not programmed correctly */
      STM_EVAL_LEDOn(LED5);
    }
    else
    {
      /* Data programmed correctly */
      STM_EVAL_LEDOn(LED4);
    }

    Address = Address + 4;
  }
```

The scary instruction:
Cast to a pointer
Pointer is dereferenced

FLASH – Domande

- 4.a Come mai nel codice di esempio viene utilizzato: *Address = Address + 4;* e non *Address = Address + 1;*?
- 4.b Quali alter funzioni s=di libreria sono disponibili per scrivere la memoria FLASH, e in cosa differiscono?
- 4.c (opzionale) Quanta memoria FLASH occupa il programma?
    - Suggerimento: e' possibile vedere nella memory window fino a quale indirizzo I settori sono diversi da 0xFFFFFFFF

FLASH – Esercizi

4.1 Controlla la correttezza dei dati scritti usando il debugger (usare Window->Memory Browser e mandare uno screenshot)

4.2 Scrivere un programma che in un intervallo 30 secondi conta il numero di pressioni del bottone, poi scrive in flash il risultato. Successivamente viene letto dalla flash il numerodi pressioni, se inferiore a 5 viene acceso il led verde, altrimenti il led rosso. (facoltativo)

4.3 Modificare il codice per scrivere in memoria una stringa si caratteri, un byte alla volta