Laboratorio di Architetture e Programmazione dei Sistemi Elettronici Industriali

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

Simone Benatti <simone.benatti@unibo.it>

Filippo Casamassima<filippo.casamassima@unibo.it>

#3 Buttons and EXTernal Interrupts

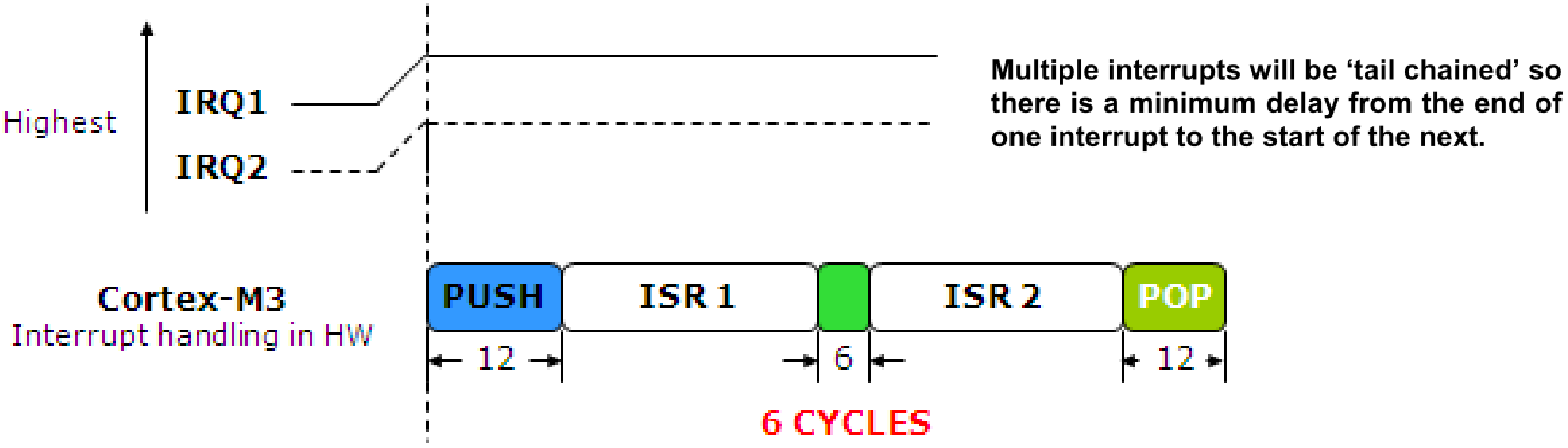
Interrupts

The Nested Vector Interrupt Controller (NVIC):

- facilitates low-latency exception and interrupt handling
- controls power management
- implements System Control Registers.

NVIC supports up to 240 dynamically reprioritizable interrupts each with up to 256 levels of priority.
There are an additional 15 interrupt sources within the Cortex core.
The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

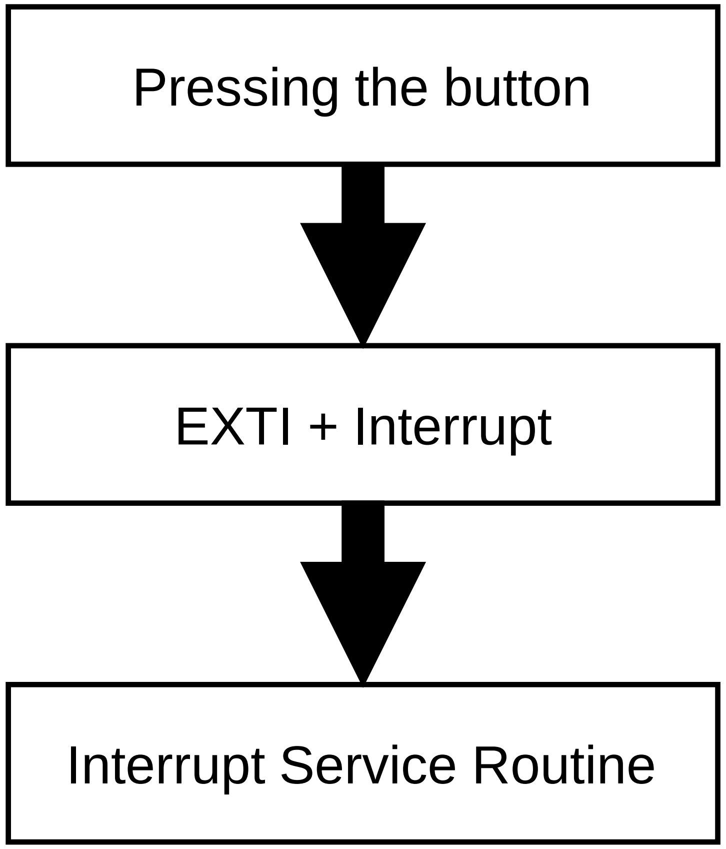
Although the NVIC is a standard unit within the Cortex core, in order to keep the gate count to a minimum the number of interrupt lines going into the NVIC is configurable when the microcontroller is designed. The STM32 NVIC has been synthesized with a maximum of 43 maskable interrupt lines.



Multiple interrupts will be ‘tail chained’ so there is a minimum delay from the end of one interrupt to the start of the next.

EXTI

- 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.



Pressing the button

EXTI + Interrupt

Interrupt Service Routine

EXTI (**what**)

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

Which bus LEDs and the button are connected to?

➡GPIO ports are always on the APB2 bus

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

➡Green LED is connected to the I/O Port C of STM32F100RB

➡Blue LED is connected to the I/O Port C of STM32F100RB

➡The button is connected to the I/O Port A of STM32F100RB

Which PINs the LEDs and the button are connected to?

➡Green LED is connected to the pin 9 of Port C

➡Blue LED is connected to the pin 8 of Port C

➡The button is connected to the pin 0 of Port A

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

EXTI (**how**)

- I want to turn on a LED using the button. **How can I do that?**

We need to setup LEDs as seen before in exercise #1

We need to setup the button. 3 steps needed.

1. Configuration of the GPIO for the button (button is connected to a GPIO)
2. Configuration of the EXTI line (because we want to use the button as a trigger for an external interrupt)
3. Configuration of the NVIC (because we want to call an ISR when button is pressed)

GPIO configuration

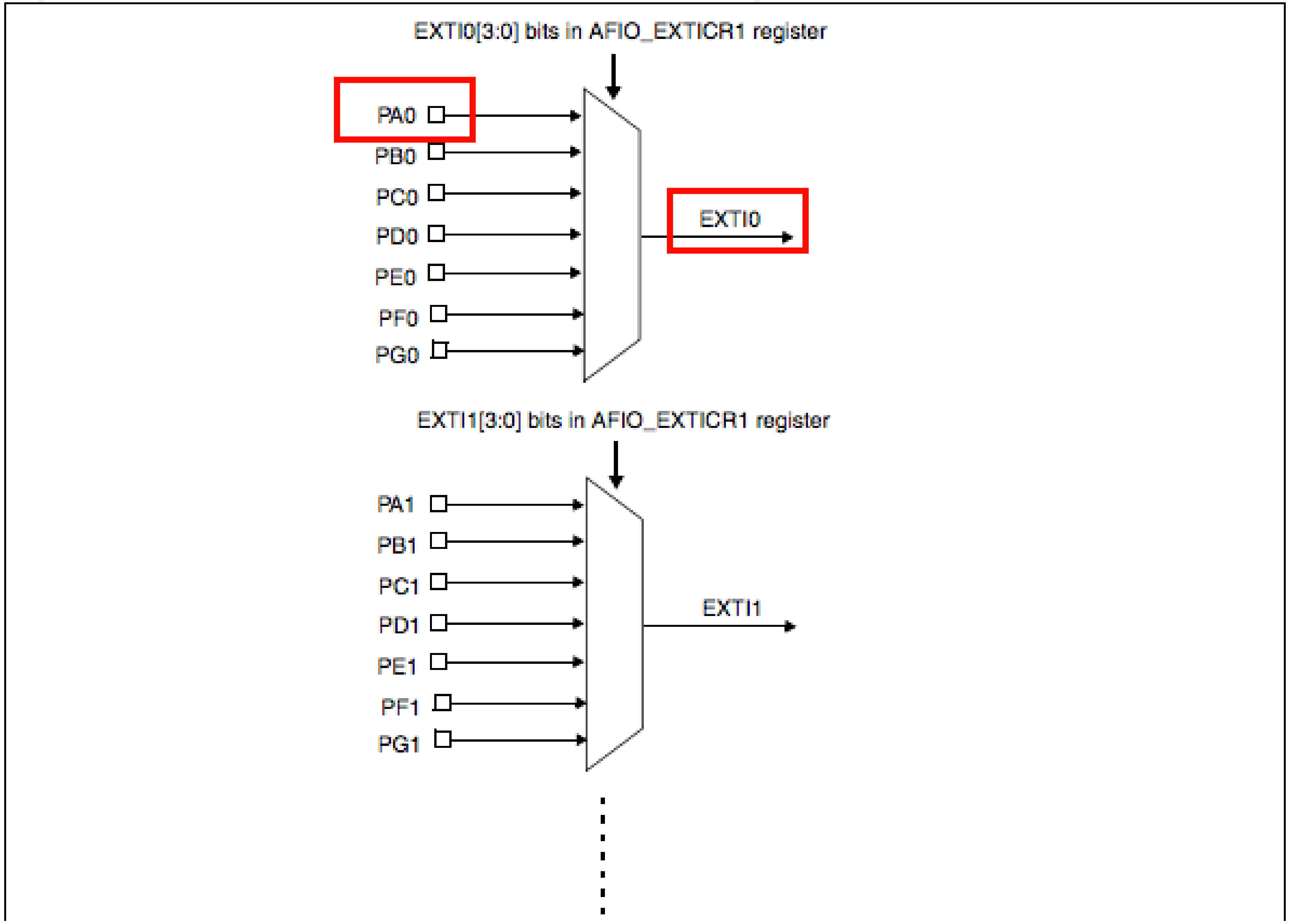
✓ Look at: stm32f10x_gpio.c / stm32f10x_gpio.h

```
GPIO_InitTypeDef GPIO_InitStructure;

/* Enable the BUTTON Clock */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

/* Configure Button pin as input floating */
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

EXTI (**how**)

- The external interrupt/event controller consists of up to 20 **edge detectors for generating event/interrupt requests**. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests

Figure 21. External interrupt/event GPIO mapping


EXTI0[3:0] bits in AFIO_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
EXTI0
EXTI1[3:0] bits in AFIO_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
EXTI1

EXTI (**how**)

EXTI line configuration

✓ Look at: stm32f10x_exti.c / stm32f10x_exti.h

```
EXTI_InitTypeDef EXTI_InitStructure;

RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

/* Connect Button EXTI Line to Button GPIO Pin */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);

/* Configure Button EXTI line */
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);
```

NVIC configuration

✓ Look at: misc.c / misc.h

```
NVIC_InitTypeDef NVIC_InitStructure;

/* Enable and set Button EXTI Interrupt to the lowest priority */
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0F;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0F;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

We need to setup the ISR

➡ The name of the EXTI0 IRQ is `void EXTI0_IRQHandler(void)`

EXTI (**code**)

main.c

```
#include "stm32f10x.h"
#include "stm32f10x_conf.h"
```

Decomment: stm32f10x_gpio.h / stm32f10x_exti.h / misc.h

```
void LEDs_Setup(void)
{
        GPIO_InitTypeDef  GPIO_InitStructure;

        /* Enable the GPIO_LED Clock */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);

        /* Configure the GPIO_LED pin */
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOC, &GPIO_InitStructure);
}

void BUTTON_Setup(void)
{
        GPIO_InitTypeDef GPIO_InitStructure;
        EXTI_InitTypeDef EXTI_InitStructure;
        NVIC_InitTypeDef NVIC_InitStructure;

        /* Enable the BUTTON Clock */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);

        /* Configure Button pin as input floating */
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        /* Connect Button EXTI Line to Button GPIO Pin */
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);
...
```

EXTI (**code**)

main.c

```
...
    /* Configure Button EXTI line */
    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);

    /* Enable and set Button EXTI Interrupt to the lowest priority */
    NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0F;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

int main(void)
{
    LEDs_Setup();
    BUTTON_Setup();    while (1);
}
```

stm32f10x_it.c

```
...

void EXTI0_IRQHandler(void){
    if(EXTI_GetITStatus(EXTI_Line0) != RESET)  {
          /* Turn ON LED3 */
                    GPIO_SetBits(GPIOC, GPIO_Pin_9);

                    /* Clear the User Button EXTI line pending bit */
                              EXTI_ClearITPendingBit(EXTI_Line0);
    }
}
...
```

EXTI (**exercises**)

3. **EXTI Exercises:**
   a) **Write a code to avoid bouncing** (google “button debounce”)

   b) **Write a code to toggle the status of the LED** (press button -> LED ON, press button again -> LED OFF, and so on...)

   c) **Write a program in which the button starts and stops the blinking of the led**

   d) **Write a program to modify the frequency of LED blinking by pressing the button** (i.e. switch ON-> LED blinks at 1 Hz / Press the button -> LED blinks at 5 Hz / Press the button -> LED blinks at 10 Hz ...)

   e) **Write a program to recognize double click on the button to turn ON the LED.** If the frequency of the clicking is too low the program should not recognize the double click. The behavior is similar to that one of a mouse: double click is recognized only if the frequency of the two clicks is high enough.

   f) **Write a code to avoid debouncing that uses the systick as a debounce timer (do not use delays in the interrupt routine)**

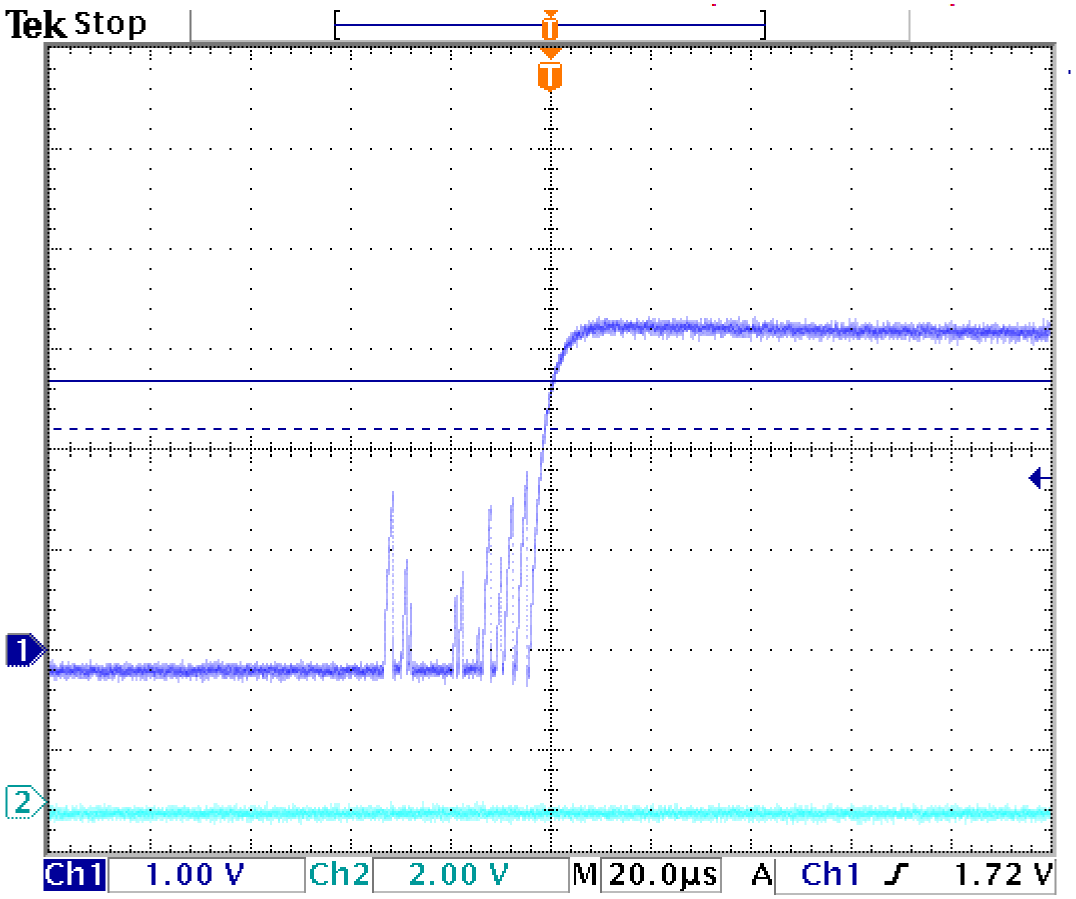
   g) **Write a code to detect interrupt on PC6, and test connecting the pin to GND. (tip: configure the line as pull up, find the name of the interrupt function in startup file)**

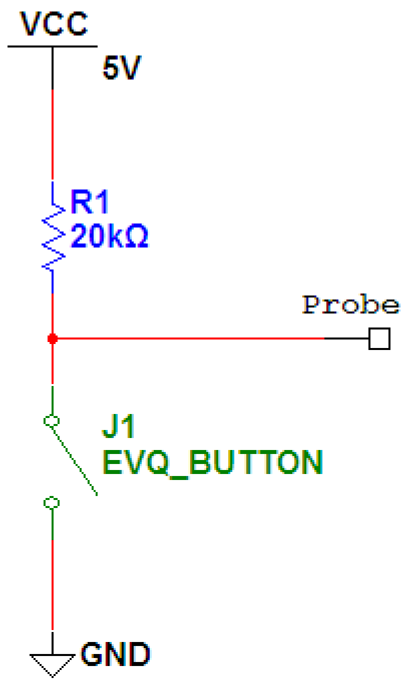
EXTI (**questions**)

- Look at the **RM0008 Reference manual** and standard peripheral library code (and also using google)
  - Why do we *set GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING* for the button?
  - Why *EXTI_InitStructure.EXTI_Line = EXTI_Line0;* ?
- What happens if we set *EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;* ?
  - Which register (look also on RM0008) are used to demine if the trigger is on the Rising/Falling edge?
  - Which parameter on *EXTI_InitStructur*e must be used to activate both Rising and Falling edge?

EXTI (**Debounce Backup**)

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.

#5 Flash memory

FLASH

- 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 written as P/E cycles)

FLASH (**what)**

- I want to write data in flash memory. **What do I need to know?**

We need to know the starting address and the end address of the memory we are going to write to

➡IMPORTANT: in flash there is also your program. So be careful: the addresses you are going to use MUST be different from the addresses where your program is.

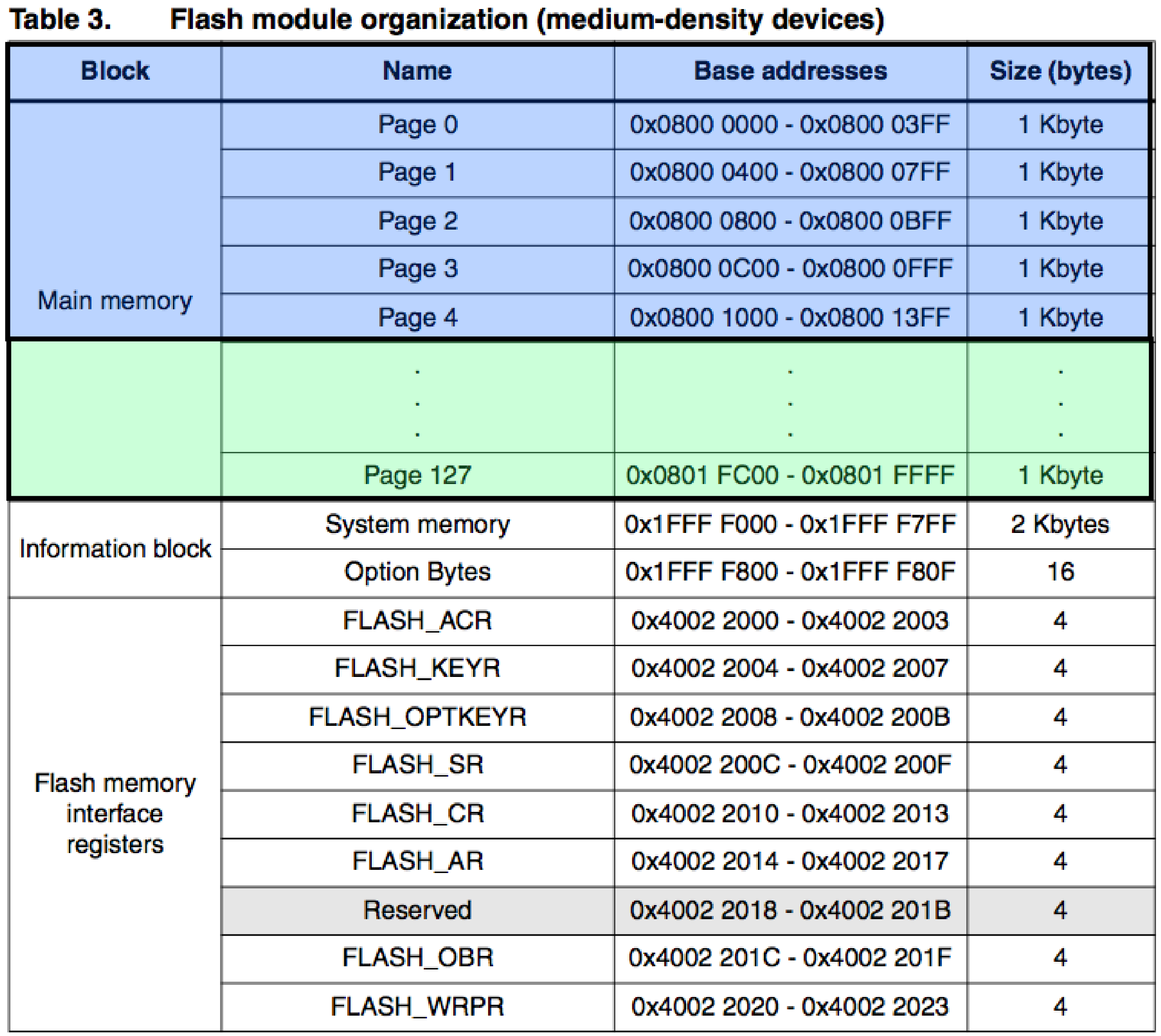


Table 3. Flash module organization (medium-density devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Your Program

Available Space

FLASH (**where**)

- I want to write data in flash memory. . **Where can I gather the information I need?**

  ➡ The **datasheet and reference manual** contain all the information we need

  ➡ Look at the **RM0041 Reference manual**

  ✓ pag.42 (medium-density devices)➠ we learn about the Flash module organization

  ✓ (In the low pages there is our program, so we can safely choose addresses from **0x08008000** to **0x0800C000**)

FLASH (**how**)

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

Unlock the Flash Bank1

➡Using void FLASH_UnlockBank1(void);

✓ Look at: stm32f10x_flash.c

Clear All pending flags

➡ Using void FLASH_ClearFlag(uint32_t FLASH_FLAG);

✓ Look at: stm32f10x_flash.c

Erase the FLASH pages

➡ Using FLASH_Status FLASH_ErasePage(uint32_t Page_Address);

✓ Look at: stm32f10x_flash.c

Program Flash Bank1

➡ Using FLASH_Status FLASH_ProgramWord(uint32_t Address, uint32_t Data);

✓ Look at: stm32f10x_flash.c

Lock the Flash Bank1

➡ Using void FLASH_LockBank1(void);

✓ Look at: stm32f10x_flash.c

FLASH (**code**)

main.c

```
#include "stm32f10x.h"

#define FLASH_PAGE_SIZE    ((uint16_t)0x400)
#define BANK1_WRITE_START_ADDR  ((uint32_t)0x08008000)
#define BANK1_WRITE_END_ADDR    ((uint32_t)0x0800C000)

uint32_t EraseCounter = 0x00, Address = 0x00;
uint32_t Data = 0x3210ABCD;
__IO uint32_t NbrOfPage = 0x00;
volatile FLASH_Status FLASHStatus = FLASH_COMPLETE;

int main(void){

        /* Unlock the Flash Bank1 Program Erase controller */
        FLASH_UnlockBank1();

        /* Define the number of page to be erased */
        NbrOfPage = (BANK1_WRITE_END_ADDR - BANK1_WRITE_START_ADDR) / FLASH_PAGE_SIZE;

        /* Clear All pending flags */
        FLASH_ClearFlag(FLASH_FLAG_EOP | FLASH_FLAG_PGERR | FLASH_FLAG_WRPRTERR);

        /* Erase the FLASH pages */
        for(EraseCounter = 0; (EraseCounter < NbrOfPage) && (FLASHStatus == FLASH_COMPLETE); EraseCounter++)    {
                                        FLASHStatus = FLASH_ErasePage(BANK1_WRITE_START_ADDR + (FLASH_PAGE_SIZE * EraseCounter));
        }

        /* Program Flash Bank1 */
        Address = BANK1_WRITE_START_ADDR;

        while((Address < BANK1_WRITE_END_ADDR) && (FLASHStatus == FLASH_COMPLETE))    {
                FLASHStatus = FLASH_ProgramWord(Address, Data);
                Address = Address + 4;
        }
        FLASH_LockBank1();

        while(1);
}
```

FLASH (**exercises**)

1. **Check the correctness of the written data using the debugger** (memory window)

- **Check the correctness of the data using code**

  ➡ Tip: you can access the flash memory using the pointers (Memory Mapping I/O)

  ➡ ((*(__IO uint32_t *) Address) != Data)

- **Write a program to store in flash memory the number of times a button is pressed in a unit of time. After this time a LED is turned ON if this number is above a threshold** (i.e. a LED is turned ON if the button is pressed more than 2 times in 5 seconds)

- **Modify the code to write in FLASH the half-word 0xBEEF and check the correctness of the data**

  ➡ Tip: FLASH_ProgramWord() cannot be used anymore.

FLASH (**questions**)

1. Look at the **RM0008 Reference manual** and standard peripheral library code (and also using google)

    ➡Why do we use Address = Address + 4 ?

    ➡What portion of the FLASH memory is used by your program?

- **Explain the meaning of the flags** FLASH_FLAG_EOP | FLASH_FLAG_PGERR | FLASH_FLAG_WRPRTERR