Laboratorio di Architetture e Programmazione dei Sistemi Elettronici Industriali

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

Simone Benatti <simone.benatti@inibo.it>

Filippo Casamassima<filippo.casamassima@unibo.it>

For Help

- At the beginning of every lab session you are provided with all the documents needed to understand what it is going on in the lab and to successfully do your homeworks

- In general:

  - if you don’t know **how to use a peripheral:** see the *STM32F10x Standard Peripherals Firmware Library.*

  - if you don’t know **how to perform something:** see the examples in *STM32F10x Standard Peripherals Firmware Library* in the section “STM32F10x_StdPeriph_Lib_Examples”.

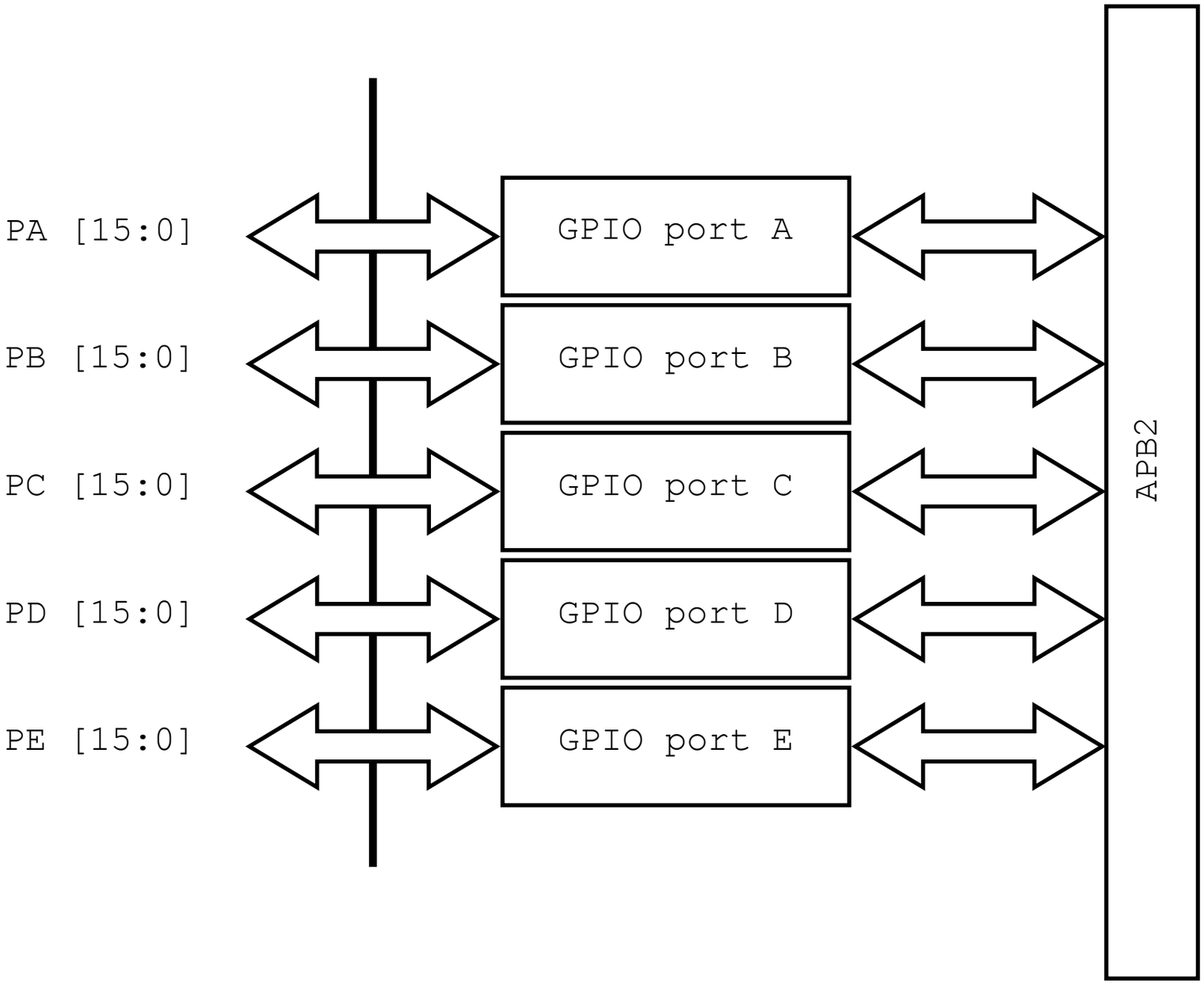
  - if you don’t know **something related to the hardware**: see the *RM0041 STM32F100xx advanced ARM-based 32-bit MCUs*.

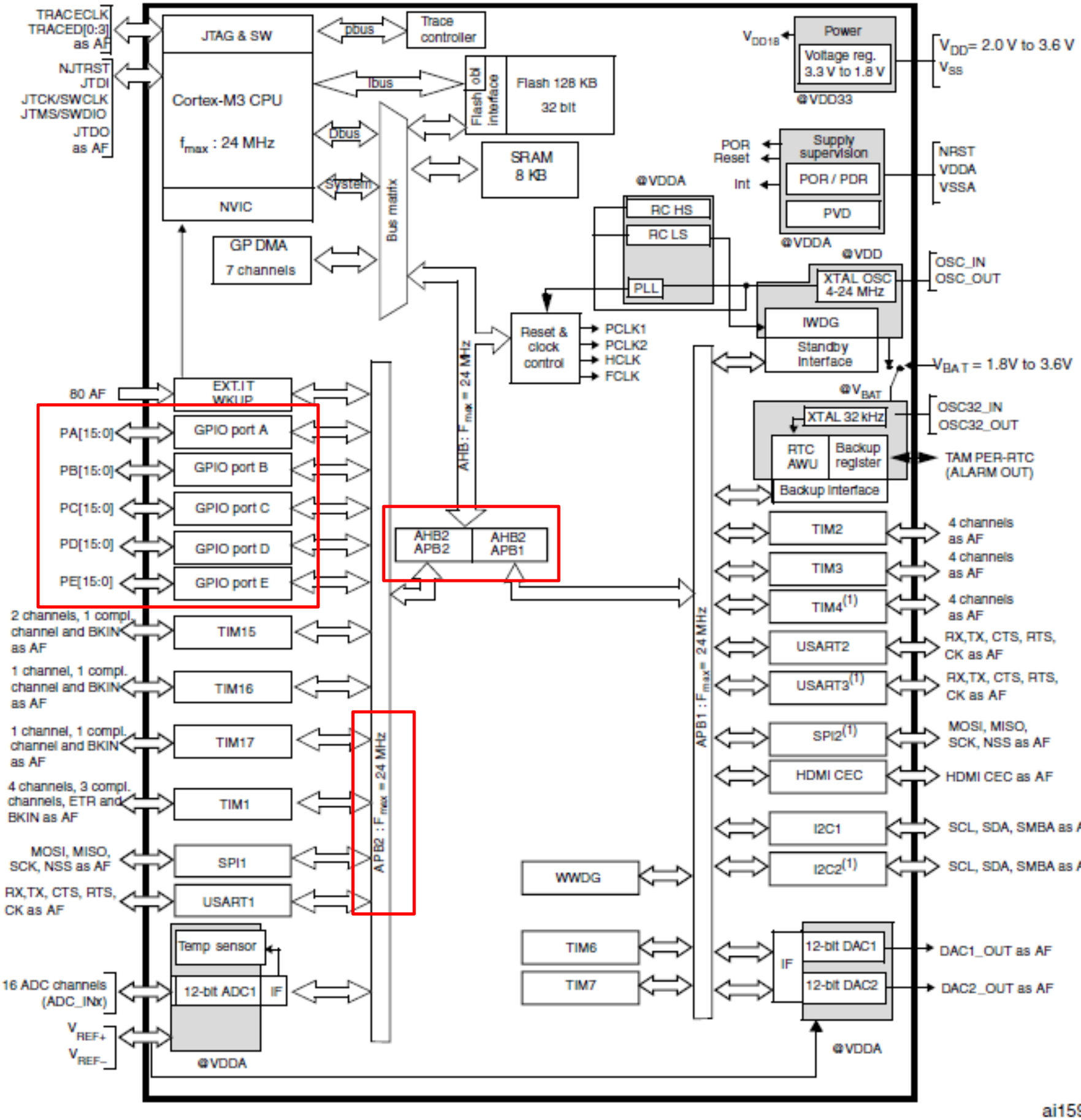
  - if **you are panicking**: google is your friend.

#1 Turn On a LED

General Purpose IO

- The STM32 is well served with general purpose IO pins, having up to 80 bidirectional IO pins. The IO pins are arranged as five ports each having 16 IO lines.




TRACECLK
TRACED[0:3]
as AF
NJTRST
JTDI
JTCK/SWCLK
JTMS/SWDIO
JTDO
as AF
JTAG & SW
Cortex-M3 CPU
f_{max} : 24 MHz
NVIC
dbus
Trace controller
Ibus
Dbus
System
Flash obl interface
Flash 128 KB
32 bit
SRAM
8 KB
Bus matrix
GP DMA
7 channels
V_{DD18}
Power
Voltage reg.
3.3 V to 1.8 V
@VDD33
V_{DD}= 2.0 V to 3.6 V
V_{SS}
POR
Reset
Int
Supply supervision
POR / PDR
PVD
@VDDA
NRST
VDDA
VSSA
@VDDA
RC HS
RC LS
PLL
@VDD
XTAL OSC
4-24 MHz
OSC_IN
OSC_OUT
IWDG
Standby interface
V_{BAT} = 1.8V to 3.6V
@V_{BAT}
XTAL 32 kHz
OSC32_IN
OSC32_OUT
RTC
AWU
Backup register
TAMPER-RTC
(ALARM OUT)
Backup interface
Reset & clock control
PCLK1
PCLK2
HCLK
FCLK
AHB: F_{max} = 24 MHz
80 AF
EXT.IT
WKUP
PA[15:0]
GPIO port A
PB[15:0]
GPIO port B
PC[15:0]
GPIO port C
PD[15:0]
GPIO port D
PE[15:0]
GPIO port E
AHB2
APB2
AHB2
APB1
TIM2
4 channels
as AF
TIM3
4 channels
as AF
TIM4(1)
4 channels
as AF
USART2
RX,TX, CTS, RTS,
CK as AF
USART3(1)
RX,TX, CTS, RTS,
CK as AF
SPI2(1)
MOSI, MISO,
SCK, NSS as AF
HDMI CEC
HDMI CEC as AF
I2C1
SCL, SDA, SMBA as AF
I2C2(1)
SCL, SDA, SMBA as AF
APB1 : F_{max}= 24 MHz
2 channels, 1 compl.
channel and BKIN
as AF
TIM15
1 channel, 1 compl.
channel and BKIN
as AF
TIM16
1 channel, 1 compl.
channel and BKIN
as AF
TIM17
4 channels, 3 compl.
channels, ETR and
BKIN as AF
TIM1
MOSI, MISO,
SCK, NSS as AF
SPI1
RX,TX, CTS, RTS,
CK as AF
USART1
APB2 : F_{max} = 24 MHz
WWDG
TIM6
TIM7
Temp sensor
16 ADC channels
(ADC_INx)
12-bit ADC1
IF
V_{REF+}
V_{REF-}
@VDDA
IF
12-bit DAC1
12-bit DAC2
DAC1_OUT as AF
DAC2_OUT as AF
@VDDA
ai15901b

General Purpose IO (**what**)

- I want to use a GPIO. **What do I need to know?**

  Which bus GPIOs are connected to?

  ➞ GPIO ports are always on the APB2 bus

  Which port are we going to use?

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

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

  Which PINs the LEDs 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

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

  ➞ I need to write (output)

General Purpose IO (**where**)

- I want to use a GPIO. **Where can I gather these information?**

  ➞ The **datasheet** contains all the information we need

  ➞ Look at the **UM0919 User Manual**

  https://www1.elfa.se/data1/wwwroot/assets/datasheets/STM32_discovery_eng_manual.pdf

  ✓ we learn about the bus APB2

  ✓ all the information we need about our LEDs

General Purpose IO (**how**)

- I want to use a GPIO. **How can I use this information to actually turn on a LED?**

We need to enable the High Speed APB (APB2) peripheral.

➡ `void RCC_APB2PeriphClockCmd(uint32_t RCC_APB2Periph, FunctionalState NewState);`

✓ Look at: `stm32f10x_rcc.c`

We need to configure the GPIO Port

➡ Fill up a `GPIO_InitTypeDef` structure

✓ Look at: `stm32f10x_gpio.h`

➡ Init the GPIO Port with `void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);`

✓ Look at: `stm32f10x_gpio.c`

Turn ON the LED

➡ `void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`

✓ Look at: `stm32f10x_gpio.c`

General Purpose IO (**code**)

main.c (green LED)

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

int main(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);

   /* Turn ON */
   GPIO_SetBits(GPIOC, GPIO_Pin_9);

   while(1);
}
```

must include stm32f10x_gpio.h

Enable APB2 bus Port C

Configuration for Pin 9 Port C as output

“And light was made”

General Purpose IO (**exercises**)

1. Turn ON the BLUE LED

- ➞ Remember: the LED is connected to the Pin 8 of the Port C (on APB2 bus)

2. Turn ON both the LEDs

3. Make the LEDs blink together

- ➞ Tip: use an empty for cycle to create a delay routine

4. Make the LEDs alternately blink

General Purpose IO (**questions**)

1. Look at the **Reference manual** and `stm32f10x_gpio.h / stm32f10x_gpio.c` (and also using google)

   ➞ Why do we set GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP ?

   ➞ Can you indicate the other modes a GPIO can be configured?

   ➞ What do “Input pull-up” and “Input pull-down” mean for the GPIO configuration? In which case are these modes useful?

#2 SysTick and toggling LEDs

SysTick

- SysTick is used to schedule **periodic** events
- When the SysTick expires an **IRQ handler** is called

SysTick (**how**)

- I want to schedule a periodic event. **How can I use SysTick?**

We need to setup the SysTick

- ➞ `static __INLINE uint32_t SysTick_Config(uint32_t ticks)`
- ➞ `ticks` is the number of ticks between two interrupts
- ➞ `SystemCoreClock` is the number of ticks in 1 sec
  - ✓ Look at: `core_cm3.h`

We need to setup the callback (Interrupt Service Routine)

- ➞ The ISR is always define in `stm32f10x_it.c`
- ➞ The name of the ISR for SysTick is `void SysTick_Handler(void)`
- ➞ Here is the code executed every `ticks` ticks

SysTick (**code**)

main.c

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

int main(void)
{
    if (SysTick_Config(SystemCoreClock / 1000))    {
        /* Capture error */
        while (1);
     }

  while (1);
}
```

ISR executed every 1 ms

stm32f10x_it.c

```
...

void SysTick_Handler(void){
    /* Here goes the code to periodically execute */}

...
```

SysTick (**exercises**)

1. **Make the LEDs blink using the SysTick**

2. **Make the LEDs alternately blink using the SysTick**

3. **Make the LEDs that blik at diferrent frequency**

4. **Make the LEDs alternately blink with a frequency lower than 1Hz.**
   - The SysTick does not support a frequency lower than 1 Hz.

2. Look at the code (not just your code, everywhere):

   ➞ What SystemCoreClock is?

   ➞ What is its value?

   ➞ Why must the ISR be named SysTick_Handler ?