LABORATORIO DI ARCHITETTURE E PROGRAMMAZIONE DEI SISTEMI ELETTRONICI INDUSTRIALI

Laboratory Lesson 2:
- General Purpose IO
- SysTick Timer

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

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

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 homework

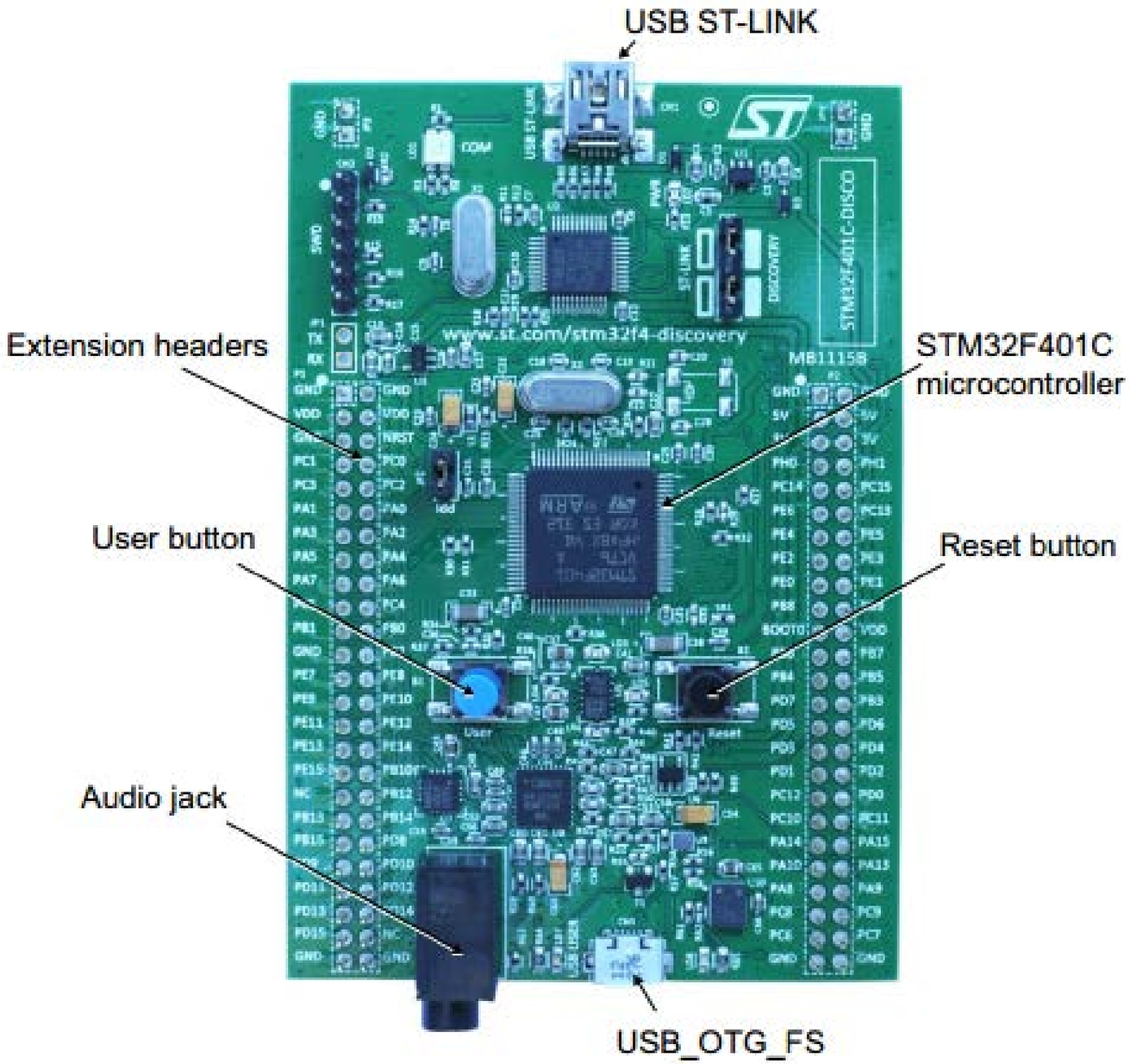
For Help

- If you do not know **how to use a peripheral:** see the *STM32F4xx Standard Peripherals Firmware Library.*

- If you do not know **how to perform something:** see the examples in *STM32F4xx Standard Peripherals Firmware Library* in the section “STM32F4xx_StdPeriph_Examples” or use the User Manual: *UM1581 User manual Description of STM32F30xx/31xx Standard Peripheral Library*

- if you don’t know **something related to the hardware**: see the Reference manual*: RM0368 - STM32F401xB/C and STM32F401xD/E advanced ARM®-based 32-bit MCUs*

STM32F401 Discovery Kit



References:

- STM32F401xB STM32F401xC datasheet

- STM32F40xxx advanced ARM®-based 32-bit MCUs reference manual (RM0344)

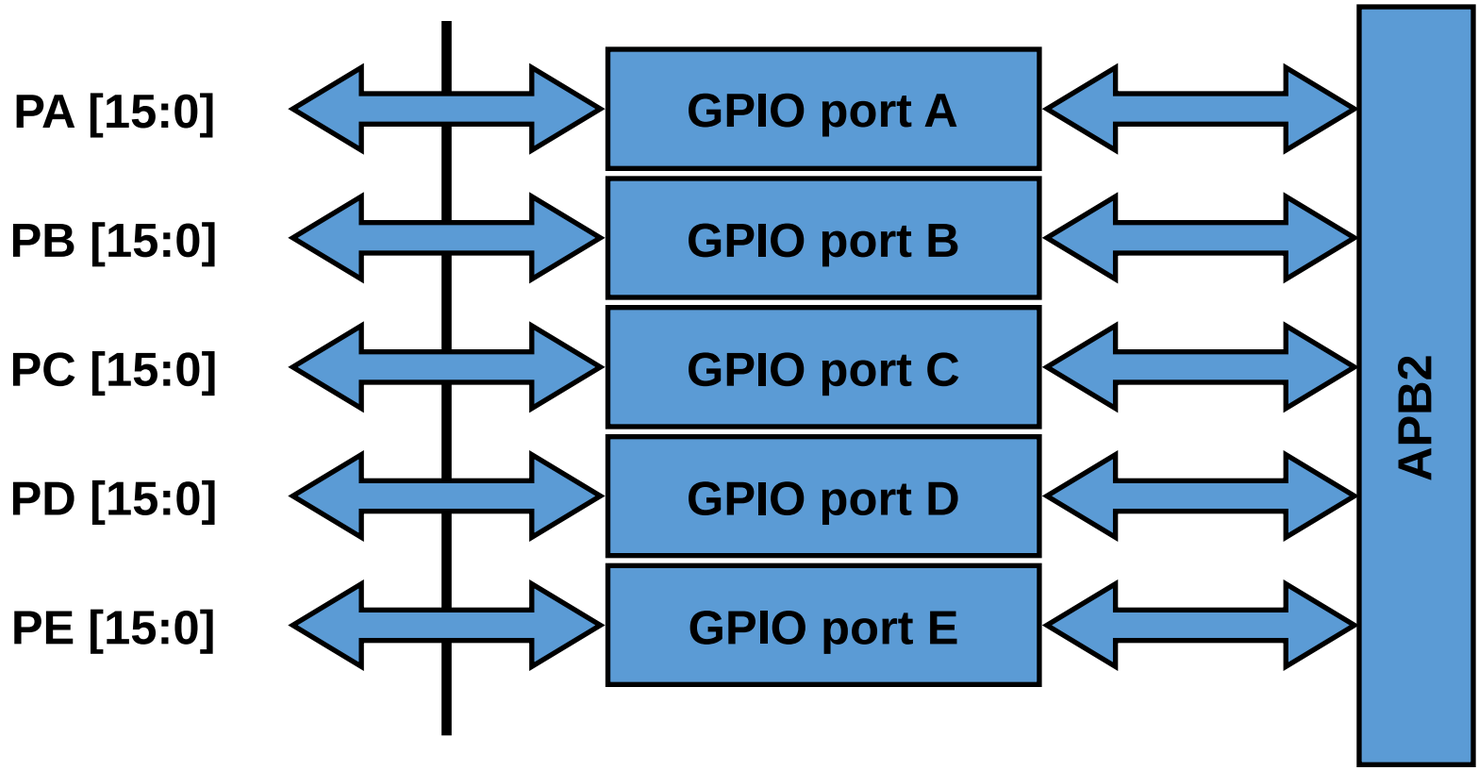
- Discovery kit for STM32F401 line (UM1669)

- Getting started with STM32F401 Discovery software development tools (UM1671)

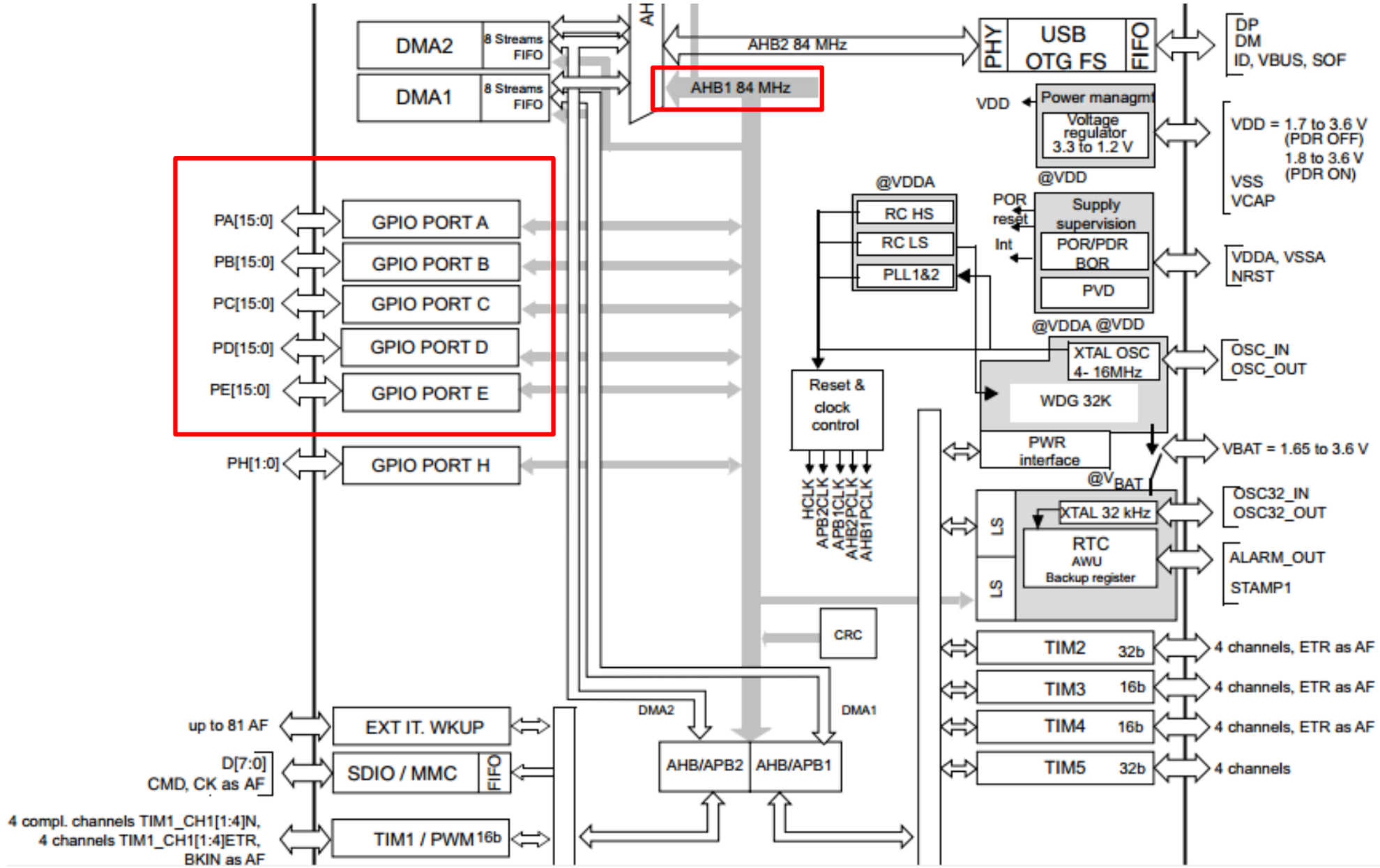
1. Turn On a LED

General Purpose IO

- The STM32 is well served with general purpose IO pins, having up to 81 bidirectional IO pins with interrupt capability. The IO pins are arranged as five ports each having 16 IO lines.



General Purpose IO Block Diagram



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 on the AHB1 bus

  - **Which PINs the LEDs are connected to?**

    Four user LEDs: LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue):

    ➡**User LD4:** The green LED is a user LED connected to the I/O PD12

    ➡**User LD3:** The orange LED is a user LED connected to the I/O PD13

    ➡**User LD5:** The red LED is a user LED connected to the I/O PD14

    ➡**User LD6:** The blue LED is a user LED connected to the I/O PD15

  - **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 **UM1660 User Manual**

http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00092826.pdf

✓we learn about the bus AHB1

✓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_AHB1PeriphClockCmd(uint32_t RCC_AHB1Periph_GPIOD, FunctionalState NewState);
      ✓ Look at: stm32f4xx_rcc.c
  - **We need to configure the GPIO Port**
    ➡Fill up a GPIO_InitTypeDef structure
      ✓ Look at: stm32f4xx_gpio.h

    ➡Init the GPIO Port with void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);
      ✓ Look at: stm32f4xx_gpio.c
  - **Turn ON the LED**
    ➡ void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
      ✓ Look at: stm32f4xx_gpio.c

General Purpose IO (code:main.c)

```
#include "stm32f4xx.h"
#include "stm32f401_discovery.h"

int main(void) {

GPIO_InitTypeDef GPIO_InitStructure; /* Enable the GPIO_LED Clock */
/
* Configure the GPIO_LED pin and enable  AHB1 BUS Port D */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

/*Configure for Pin 12 Port D as output*/
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);

GPIO_SetBits(GPIOD,GPIO_Pin_12); /*And light was made*/
for(;;); }
```

General Purpose IO (esercizi)

1.1 Accendere il LED BLU

1.2 Accendere entrambi I LED: VERDE e BLU

1.3 Scrivere un codice che faccia lampeggiare tutti I led insieme: VERDE, BLU, ARANCIONE, ROSSO.

1.4. Fare lampeggiare I Led in maniera alternata: VERDE, BLU, ARANCIONE, ROSSO.

General Purpose IO (domande)

Tenendo come riferimento il **Reference manual,** stm32f10x_gpio.h/stm32f4xx_gpio.c e Google:

1.A Perchè non configuriamo GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP ?

1.B Descrivere in dettaglio le altre modalità con cui I GPIO possone essere configurati

1.C Che significato hanno le configurazione “Input pull-up” e “Input pull-down” ed in quali casi sono utili?

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

  - **We need to setup the callback (Interrupt Service Routine)**
    - ➡ The ISR is always define in stm32f4xx_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 "stm32f4xx.h"
#include "stm32f4xx_conf.h"

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

  while (1);
}
```

ISR executed every 1 ms

stm32f4xx_it.c

...

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

SysTick (Esercizi)

2.1 Fai lampeggiare i Led utilizzando il SysTick

2.2 Fai lampeggiare I led alternativamente utilizzando il SysTick

2.3 Fai lampeggiare due led a differente frequenza utilizzando per entrambi il SysTick

2.4 Fai lampeggiare due led alternativamente con una frequenza inferiore a 1Hz: ricorda che il Systick non supporta frequenza inferiori ad 1Hz.

SysTick (Domande)

Osservando il codice, non solo quello implementato da voi ma più in generale:

2.A Che cos'è SystemCoreClock?

2.B Qual è il suo valore?

2.C Qua'è il ruolo della ISR chiamata SysTick_Handler ?