–Elettronica dei Sistemi Digitali –
Lab#7

Light Emitting Diode,
Pushbutton and Square Waves

The purpose of this laboratory is to learn how to drive the light-emitting diode (LED) and use the pushbutton available on the NUCLEO board. Detailed information on how to program the board could be found either on the lab documentation available on the website or on the reference manual of the board. For each exercise, you have to follow all the assignments. If some questions are present at the end of the assignments, be ready to answer them in the lab report.
The numbers near each exercise refer to specific sections of the documentation. You should read all the documentation file before starting to write your code!

Contents:

Abbreviations and acronyms:

LED	– Light Emitting Diode
SW	– Switch
MCU	– Microcontroller Unit

0 - Program example (7.1, 7.2, 7.3, 7.4)

In the following, we show a complete example of a C-program intended to turn on and off the board LED. The example has been developed using a low-level approach, thus, writing to the registers directly. The required steps are the following:

1. **Understand** the code, full description can be found in the documentation.
2. **CAREFULLY READ THE DOCUMENTATION**
3. **Create a new project** using the “STM32CubeIDE” (empty project version).
4. **Copy and paste** the code below inside the main file of the new project (overwrite all the default code).
5. **Compile** the project.
6. **Download** the compiled code on the Nucleo board.
7. **Test** the functionality of the circuit by looking at the LED.

Once the code has been verified, you have to set a breakpoint at line 29. Then use the debugger to follow the next points.

1. **Run the debugger**.
2. Use the proper window to **read the value of the registers** related to the LED.
3. **Step one instruction.**
4. **Check again the same registers** and find the differences.
5. **Repeat again from step 1 once**.

What happens in the registers? And on the board?

NB: the debugger is the ONLY way to find errors in your code. You should use the debugger to test the functionality.

```c
#define MYWAIT 1000000

int main(void)
{

        //PORT REGISTERS
        volatile unsigned int *GPIOA_MODER = (unsigned int*) (0x40020000 + 0x00);
        volatile unsigned int *GPIOA_ODR = (unsigned int*) (0x40020000 + 0x14);

        //CLOCK REGISTERS
        volatile unsigned int *RCC_AHB1ENR = (unsigned int*) (0x40023800 + 0x30);

        //VARIABLES
        int i;

        //ENABLE PORT CLOCK:
        // this ensure that the peripheral is enabled and connected to the AHB1 bus
        *RCC_AHB1ENR |= 0x01U;

        //CONFIGURE PORT: set MODER[11:10] = 0x1
        *GPIOA_MODER = *GPIOA_MODER | 0x400;

        //SWITCH ON THE LED: set ODR[5] = 0x1, that is pulls PA5 high
        *GPIOA_ODR = *GPIOA_ODR | 0x20;

        // Application code (Infinite loop)
        while (1)
        {
                 *GPIOA_ODR = *GPIOA_ODR ^ 0x20;
                for (i=0; i<MYWAIT; i++)
                {
                }
        }
}
```

1 - Controlling the LED

The NUCLEO board provides a green LED, that can be used to display output values. This LED is physically connected to a port of the processor. The LED is turned on with a '1' and off with a '0'. Furthermore, two pushbuttons are present. The black one is connected to the reset of the MCU. The blue one, on the contrary, can be used as digital input.

1.1 - Use switch to switch on/off the LED Low Level Approach (7.4)

In this first exercise you will directly manipulate register in order to configure the peripherals. You have to modify the last exercise in order to turn on the LED when the pushbutton is pressed. The required steps are the following:

1. Use **copy and paste from the project navigator** to duplicate the previous project in a new one.
2. **Modify the code** to turn on the led when the pushbutton is pressed.

3. **Compile** the project.
4. **Download** the compiled code on the Nucleo board.
5. **Test** the functionality of the circuit by looking at the LED.

1.2 - Use switch to switch on/off the LED (7.5, 7.6)

Direct manipulation of the registers is no longer the common way for programming an MCU. All the vendors have their own set of libraries and API (application programming interface) that are developed to simplify the work of the programmers. For this purpose, we will use the LL (Low Layer) developed by ST. All the exercises from now on will require you to select the proper configuration through the STM32Cube (detailed information is available on the lab documentation). In this exercise you have to develop a C project where led is on when the pushbutton on the NUCLEO board is pressed. The required steps are the following:

1. **Create a new project** using the “STM32CubeIDE”.
2. **Configure the hardware** using “STM32Cube”
3. **Fully understand the code generated** by the “STM32Cube”.
4. **Write a C program** that reads the pushbutton and turn on the LED.
5. **Compile** the project.
6. **Download** the compiled code on the Nucleo board.
7. **Test** the functionality of the circuit by looking at the LED.

1.3 - Varying blinking frequency (7.6)

Develop a C project that at each pression of the pushbutton double the blinking frequency of the LED. After the reset the LED should blink at roughly 0.25 Hz. After hitting the pushbutton, the frequency should be 0.5 Hz, then 1 Hz and so on. The required steps are the following:

1. **Create a new project** using the “STM32CubeIDE”.
2. **Configure the hardware** using “STM32Cube”
3. **Fully understand the code generated** by the “STM32Cube”.
4. **Write a C program** that reads the pushbutton and update the WAIT value in order to meet the specification.
5. **Compile** the project.
6. **Download** the compiled code on the Nucleo board.
7. **Test** the functionality of the circuit by looking at the LED.
8. **Use the debugger** to read the WAIT value after each hit of the pushbutton.

Is there a limit for the appreciable change in the frequency? Why? If yes, which is the limit value of WAIT?

2 - Generating a Square Wave (7.6)

The same approach that you have used with the LED can be adopted to generate a square wave on a general-purpose input output (GPIO) pin. You have to develop a C project to generate a square wave. The desired frequency of the waveform is 1 kHz and the PA10 pin on the Nucleo board must be used. The required steps are the following:

1. **Create a new project** using the “STM32CubeIDE”, selecting the proper configuration for pin PA10.
2. **Fully understand the code generated** by the Cube.
3. **Follow the instruction** described at the end of this document.
4. **Write a C program** that toggles the pin state with the desired frequency.
5. **Compile** the project.
6. **Download** the compiled code on the Nucleo board.
7. **Test** the functionality of the circuit by using the oscilloscope.

Now follow the pass described at the end of this exercise and perform again the test. Is the waveform stable? Do you see any strange behavior on the oscilloscope?

Some parts of this code could be unclear now, in particular the body of the function must be taken as it is without introducing any modification. The understanding of these portions is the topic of the next labs.

1. Insert the following line of code in your *main* file, just before the “while(1)”

```
SysTick_Config(SystemCoreClock / 1000);
```

2. Locate file named “stm32f4xx_it.c” in your project
3. Copy the code below in the body of the function called “void SysTick_Handler(void)”

```
static int x=0x12c; // What is this number?
for(int i = 0; i<x; i++);
x = (x >> 2) | (((x & 1) ^ (x & 2)) << 5);
```