POLITECNICO DI TORINO

–Elettronica dei Sistemi Digitali –

Lab7

Light Emitting Diode
Pushbutton and Square Waves

The purpose of this laboratory session is to learn how to drive the light emitting diode (LED) and to use the pushbutton available on the NUCLEO board. Detailed information on how to program the board could be found in the LAB7 accompanying document [1] and in the reference manual of the board. The laboratory session is divided in several projects (see the list below), which are here presented in separate Sections. For each project, you have to

- Complete all the project assignments,
- Include in your laboratory report a dedicated Section, describing the main steps followed, your choices and the achieved results,
- Deliver a separate folder with any source file edited by you (this always includes the *main.c* file; additional files could be required, but do not include the files generated automatically by the tools). Merge the whole set of folders in a single archive file including all projects in the laboratory session.

Projects:

0. - Program example: turning on and off the board LED (no deliverables for this project)
1.1 - Using a switch to switch on/off the LED, Low Level Approach
1.2 - Using a switch to switch on/off the LED, STM32Cube approach
1.3 – Varying the blinking frequency
2 - Generating a square wave

To complete these projects, you need the STM32CubeIDE toolchain and the main documents on the MCU (see the list of References in [1]). In addition, since an emulator of the STM32 is not available, you also need the Nucleo board, connected to your PC. Finally, some of the projects require the availability of a few elementary measurement instruments, such as an oscilloscope and a waveform generator. For these projects, the present document describes an alternative assignment, which opens the possibility to complete the assigned projects with no needs for laboratory instruments. The guidelines and specifications useful to complete the alternative assignments are highlighted in bright green.

Abbreviations and acronyms:

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

0 - Program example

A complete example of a C-program that turns on and off the board LED is given below. The example has been developed using a low layer approach, thus, writing the configuration registers directly, with no use of library drivers. The required steps are the following:

1. **Read and understand** the C code and the LAB7 accompanying document.
2. **Create a new project** by using the "STM32CubeIDE" (empty project version).
3. **Copy and paste** the code below inside the main file of the new project (overwrite all the default code).
4. **Compile** the project.
5. **Download** the compiled code on the Nucleo board.
6. **Test** the functionality of the circuit by looking at the LED.

```
1   #define MYWAIT 1000000
2
3   int main(void)
4   {
5
6           //PORT REGISTERS
7           volatile unsigned int *GPIOA_MODER = (unsigned int*) (0x40020000 + 0x00);
8           volatile unsigned int *GPIOA_ODR = (unsigned int*) (0x40020000 + 0x14);
9
10          //CLOCK REGISTERS
11          volatile unsigned int *RCC_AHB1ENR = (unsigned int*) (0x40023800 + 0x30);
12
13          //VARIABLES
14          int i;
15
16          //ENABLE PORT CLOCK:
17          // this ensure that the peripheral is enabled and connected to the AHB1 bus
18          *RCC_AHB1ENR |= 0x01U;
19
20          //CONFIGURE PORT: set MODER[11:10] = 0x1
21          *GPIOA_MODER = *GPIOA_MODER | 0x400;
22
23          //SWITCH ON THE LED: set ODR[5] = 0x1, that is pulls PA5 high
24          *GPIOA_ODR = *GPIOA_ODR | 0x20;
25
26          // Application code (Infinite loop)
27          while (1)
28          {
29                  // Add your code here.
30                  *GPIOA_ODR = *GPIOA_ODR ^ 0x20;
31                  for (i=0; i<MYWAIT; i++)
32                  {
33                  }
34          }
35
36  }
37
38
39
```

Once the code has been verified, you have to set a breakpoint at line 30. Than use the debugger and follow the next points.

1. **Run the debugger**.
2. On the STM32CubeIDE tool, use the proper window to **read the value of the peripheral registers** related to the LED.
3. **Step one instruction.**
4. **Check again the same registers** and find the differences.
5. **Repeat again from step 2**.

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 before calling the lab assistant.

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 - Using a switch to switch on/off the LED, Low Level Approach

In this project, you will directly manipulate the configuration registers in order to configure the peripherals. You have to modify the previous project, in order to switch on the LED while 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 switch on the LED while the pushbutton is pressed (the LED must switch off when the pushbutton is released).
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 and by running the debugger.

1.2 - Using a switch to switch on/off the LED, STM32Cube approach

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. To 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 (more information are available in [1]).
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" and understand the generated code.
3. **Write a C program** that reads the pushbutton and properly drives the LED (add your code in the user code sections of the *main.c* file generated by STM32Cube).
4. **Compile** the project.
5. **Download** the compiled code on the Nucleo board.
6. **Test** the functionality of the circuit by looking at the LED and by running the debugger.

1.3 - Varying the blinking frequency

Develop a C project that doubles the blinking frequency of the LED at each pressure of the pushbutton. After the reset the LED should blink at roughly 0.25 Hz. After pressing 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" and understand the generated code.
3. **Write a C program** that reads the pushbutton and update a wait variable in order to meet the specification.
4. **Compile** the project.
5. **Download** the compiled code on the Nucleo board.
6. **Test** the functionality of the circuit by looking at the LED.
7. **Use the debugger** to read the wait value after each pressure of the pushbutton.

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

2 - Generating a Square Wave

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 D2 pin on the Nucleo board must be used. You have to follow the steps described at the end of this Section before running your program. The required steps are:

1. **Create a new project** using the "STM32CubeIDE", selecting the proper configuration for pin D2.
2. **Configure the hardware** using "STM32Cube" and understand the generated code.
3. **Follow the instructions** 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.

Is the waveform stable? Do you see any strange behavior on the oscilloscope? Modify your main code by commenting the line inserted just before the while loop, then run again the application and compare the results.

Code to be added:

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)) << 4);
```

Project version with no need of measuring instruments.

If a laboratory oscilloscope is not available, you can complete the assigned project and display on the screen of your PC the generated waveform by exploiting the **waveform viewer** application described in the Appendix of Section 3. This application includes three parts: (i) an STM32 application, which samples the waveform and send the obtained samples to the PC, through the ST-Link connecting the PC to the Nucleo board; (ii) a terminal emulator, which receives the samples and saves them on a file; (iii) a Matlab script that plots the waveform.

Read the content of the Appendix in Section 3. The details of the STM32 application code may be partially unclear at this time, because you did not yet study all peripherals that are involved in this application. However, you should be able to follow the described procedure.

3 - Appendix – Waveform viewer

This Section briefly introduces an STM32 application developed to replace the oscilloscope with a PC based waveform viewer, which allows displaying a waveform received on an analog input pin of the Nucleo board. This application can be used to evaluate the quality of a waveform and to derive some key time and frequency values, such as period, frequency and duty cycle.

This application includes three parts:

1. An STM32 application code (viewer.rar file) to be extended with your own application code and loaded on the Nucleo board
2. A terminal emulator (RealTerm) running on the PC and connected to the board
3. A Matlab/Octave script (plots.m) to visualize the waveform.

The waveform at the analog input pin PA1 is sampled by one of the ADC channels available on the MCU; the acquired samples are then sent to the PC using the UART interface; on the PC side, a terminal application captures the stream of samples, which are saved in a file. Finally, the samples are shown graphically by using the Matlab/Octave script.

The STM32 application.

The MCU application code in contained in the related STM32CubeIDE project (available on the course web pages, in the folder viewer). This project contains all configuration code required to enable both the waveform sampling and the transmission of samples to the PC. The following units are configured:

- Timer TIM2 is configured in mode OC-TRGO, to provide the sampling trigger to the ADC.
- Analog to digital converter ADC1 is configured for acquiring 8-bit samples and send them to a 256-location memory buffer by using DMA normal mode (DMA2-Stream0).
- The USART2 is configured in asynchronous mode, with baud-rate 115200 bit/s, 8 data bits, 1 stop bit and no parity. The unit reads samples from the memory buffer in DMA normal mode (DMA1-Stream6).

The application alternates two phases: in the first one, only the ADC operates to acquire a block of 256 samples, to be saved in the buffer. At the end of phase 1, the ADC stops and an interrupt service routine enables the UART transmission. In the second phase, the 256 samples are sent to the PC and, when the transmission is complete, an interrupt service routine enables again the ADC to obtain a new sequence of samples.

The terminal.

To receive samples, you have to install the terminal application RealTerm on your PC (see for example https://sourceforge.net/projects/realterm/). Then you can launch it and configure the key options:

1. In the Port section, set the baud-rate to 115200 bit/s, 8 data bits, 1 stop bit and no parity. Moreover, enter the name of the COM port used by the ST-Link.
2. In the Display section, select uint8, to see the received samples in the integer format, or Hex, to see them in the hexadecimal notation.
3. In the Capture section, set the Hex option to capture the bytes in the hexadecimal format.

While the application is running on the Nucleo board, use the Start and Stop commands on RealTerm to capture a sequence of samples and save them in the showed file. One second or two are enough to capture enough samples in most of cases; you can use longer capture times if you want to look at very low frequency waveforms.

The Matlab script.

Finally, open Matlab and run the plots script (plots.m, available on the course web pages) to read samples from the file, convert and display them over time on a window, and finally show the measured frequency.

Procedure.

In order to display a waveform generated on a pin of the Nucleo board, as specified in some assignments (e.g. a square wave created with a timer or a sine wave produced by means of a PWM unit), you have to develop your project according to these steps:

1. Starting from the original viewer project (the one containing the waveform viewer application described above), create a new project including both the viewer code and the code required to configure and run your own application. Be aware that the existing code (including the configuration code, the interrupt service routines and the main code) must not be modified and any new code inserted by you must not interfere with the original viewer project. The suggested procedure to do that is as follows:
    a. Download the viewer project, save it in a folder and extract the contained files.
    b. From STM32CubeIDE, follow the path *File-> new -> STM32 project from STM32CubeMX .ioc file* to create a new project and import the viewer .ioc file.
    c. From STM32CubeIDE, follow the path *Project -> Generate Code* to regenerate the code.
    d. Copy the entire folders *Inc*, *Src* and *Startup* from the *Core* directory of the downloaded project to the same position in the new project.
    e. As a result of these steps, the new project is a complete copy of the original viewer project, properly configured and linked to the related drivers. By working on the new project, you can add all the elements required for the application specified in the assignment. This operation can be done by both working on the Cube perspective (.ioc file) and entering new code in the source files (e.g. main.c or interrupt service routines).
2. On the Nucleo board, use a wire to connect the output pin where the waveform is generated and the input analog pin connected to the ADC (pin 1 in the GPIO port C, corresponding to position A4 in the CN8 Arduino connector).
3. Compile and run your application.
4. Configure the RealTerm terminal as detailed above and, on the Capture section, activate the capture for a short time.By default, RealTerm saves the captured samples to a file in the c:\temp folder.
5. Copy and possibly rename the sample file from c:\temp to the same directory including the Matlab plots.m file. Then, run the script, select your sample file and look at the displayed waveform. The script allows loading and plotting multiple file, to compare different signals.

Figure 1 is a screenshot showing the mentioned applications at work.

- Left bottom corner: STM32CubeIDE window with running application code
- Right bottom corner: external waveform (sine wave generated using a waveform generator)
- Left top corner: RealTerm with configuration options and received samples
- Right top corner: Matlab window, with viewer code and plot.

Be aware that the captured samples typically include multiple blocks of 256 samples and these blocks are not adjacent. This is due to the implemented application, which alternates acquisition and transmission phases: during the transmission of a block, the ADC does not sample the waveform and when the ADC is started again, a number of samples have been skipped. Therefore, if the capture spans over more than 256 samples, several missing samples separate one block from the next one. This is clearly visible in the time domain plot, but it usually does not affect the evaluation of the main frequency component.

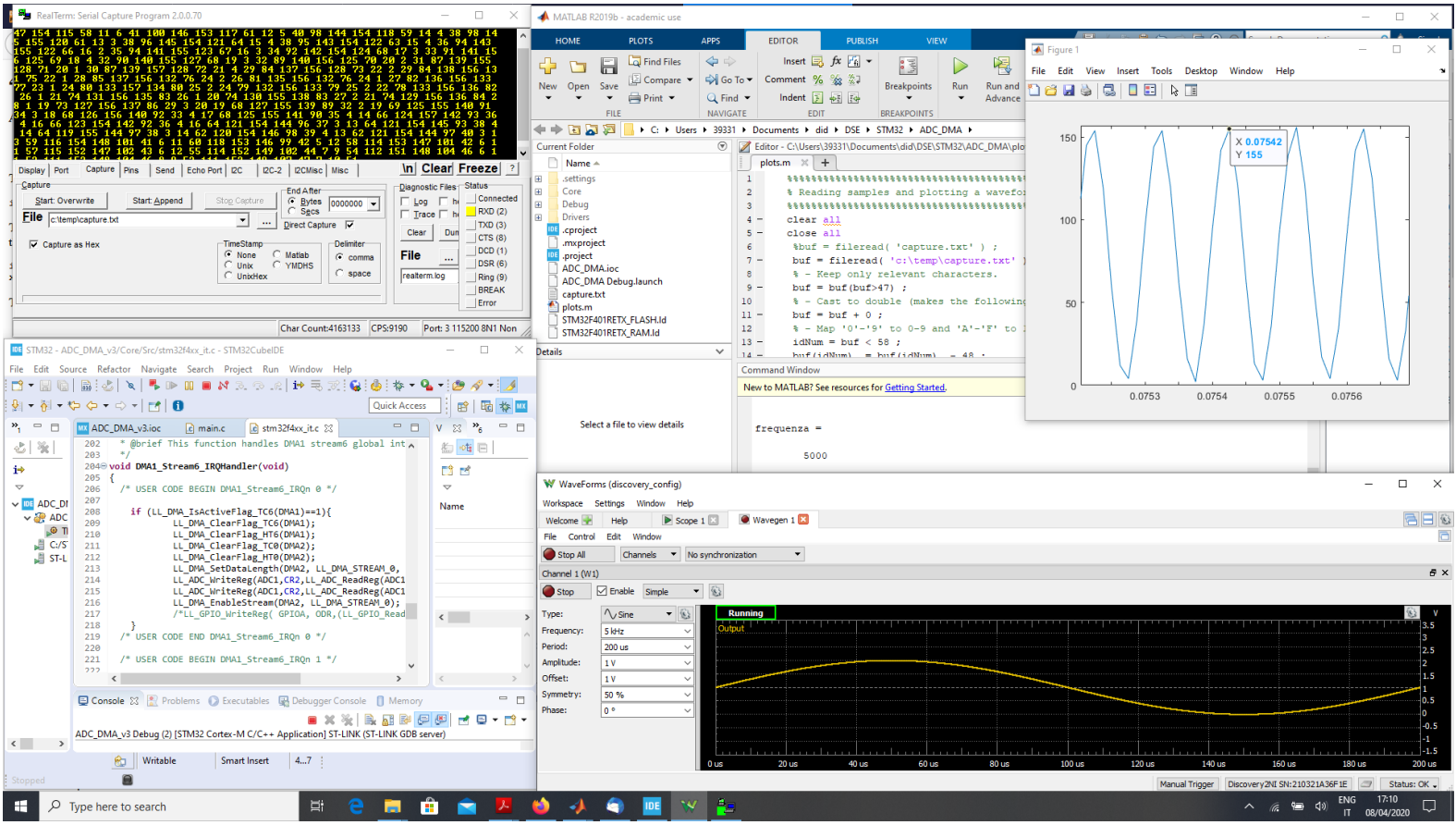

Figure 1: Waveform viewer applications.

References

[1] Lab7-STM32-document, Introduction to the STM32 toolchain and NUCLEO board Experiments with Light Emitting Diodes and Switches