Politecnico di Torino

– Elettronica dei Sistemi Digitali –
Lab#10

Input capture, Pulse Width Modulator and Serial communication using HAL

The purpose of this laboratory session is twofold:

i) to learn how to use the input compare function of the Timer;
ii) to learn using the Pulse Width Modulator (PWM);
iii) to learn using the serial communication.

In this laboratory session, you will move from LL to HAL, so all the exercises must be implemented in HAL.
Some hints on how to use HAL for the timer, are reported on /Materiale/Laboratorio/LL_vs_HAL.pdf on Portale della Didattica.
Detailed information on how to program the board could be found on the lab documentation available on the website, and on the reference manual of the board. For each exercise you must follow all the assignments. If some questions are present at the end of the assignments, be ready to answer that in the lab report.

Contents:

Abbreviations and acronyms:

LED – Light Emitting Diode
MCU – Microcontroller Unit
TIM – Timer
ADC – Analog-to-Digital Converter
PWM – Pulse Width Modulator

1- [HAL] Measure of the frequency and duty cycle of an external square waveform (10.1, 10.2)

While the output compare is used to toggle a pin when a value is reached by the counter, input capture is the opposite. When an edge is detected, the counter value is saved in a dedicated register and read later.
Write a program that measures frequency and duty cycle of an input square waveform coming from a function generator using the input capture of TIM3. **Frequency/Duty cycle measurements should be performed using the debugger.** The required steps are the following:

1. **Create a new project** using the "STM32CubeIDE".
2. **Configure the hardware** using "STM32Cube", selecting the proper configuration for counter TIM3. **In particular, set the prescaler to 99.**
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that evaluates the frequency and duty cycle from the captured values with TIM3.

5. **Write the interrupt service routine (ISR)** for TIM3.
6. **Compile** the project.
7. **Download** the compiled code on the NUCLEO board.
8. **Configure properly the function generator and connect it to the input pin**. Read carefully the documentation for the setup.
9. **Test** the functionality of the circuit by running it in debug mode and stopping the execution after the calculation of the frequency and changing the input waveforms.

- If the timer peripherals are kept enabled during debugging phase, what happens if you use the breakpoints to stop execution every cycle?

Try now to **freeze the TIM3 peripherals** by means of:

```
__HAL_DBGMCU_FREEZE_TIMx()
```

(Reference on pages 52-55 in the user manual).

- Is the system measuring the correct frequency/duty cycle? Why?
- What happens if you use the breakpoints to stop execution every cycle? Try to leave the program running for a while after setting the new input and then pause the program. Does it work? Why?
- What are the upper and lower limit for the detected input frequency? Why? What could you do to increase them?
- How could you increase the accuracy of your measurements?

2- Pulse Width Modulation

2.1- [HAL] Square wave generation (10.3)

Another way to generate a square wave is using the PWM function. PWM is simpler to configure with respect to output compare function and it is normally used to perform more complex operations. You have to use the PWM function to generate a square wave at 10kHz, using TIM4. **Start from the previous exercise and use the frequency measured by TIM3 to set up the duty cycle of the PWM.** The frequency range of the input signal must be f_{IN}= 800 Hz-12 kHz corresponding to a duty cycle ranging from 25% to 75% (800 Hz → 25% and 12 kHz → 75%). The required steps are the following:

1. **Create a new project** using the "STM32CubeIDE".
2. **Configure the hardware** using "STM32Cube", selecting the proper configuration for counter TIM3 and TIM4.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that evaluate frequency of the input value and write the correct value for the duty cycle.
5. **Write the interrupt service routine (ISR)** for TIM3.
6. **Compile** the project.
7. **Download** the compiled code on the NUCLEO board.
8. **Configure properly the function generator and connect it to the input pin**. Read carefully the documentation for the setup.
9. **Verify if the TIM4 PWM works properly by connecting the oscilloscope to the corresponding pin.**
10. **Try to change manually the duty cycle of the PWM by changing the input frequency.** What are the effects of the variable duty cycle generated by the PWM on the oscilloscope?

2.2-[HAL] LED dimmer (10.4, 10.5, 10.6, 10.7)

One task commonly performed in PWM is dimming intensity of the power transmitted to a load. You will use this approach to regulate the LED2 brightness by means of the Pulse Width Modulation (PWM) unit of TIM2 timer, since it is directly connected to the LED. Specifically, we want to gradually increase the brightness of the LED from the minimum (dark LED) to the maximum (LED fully switched on) in one second; then, we want to switch from the maximum to the minimum in a second. The two transients must be repeated indefinitely. Since we control the LED by switching it on and off, in order to avoid any visible flickering of the light, we must select a frequency of 100 Hz or more for the PWM and then change the duty cycle properly. The required steps are the following:

1. **Create a new project** using the "STM32CubeIDE".

2. **Configure the hardware** using "STM32Cube", selecting the proper configuration for counter TIM2 and the LED.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that change the value for the duty cycle following the specifications.
5. **Compile** the project.
6. **Download** the compiled code on the NUCLEO board.
7. **Test** the functionality of the circuit by looking the LED.

Use an oscilloscope to see the waveform present on the LED pin. What do you see?
Once your project works, try to modify prescaler and period values keeping the same frequency value. For example, double the period and half the prescaler. What do you see on the LED? Do you see a different behavior? Why?

3. [HAL] Control the LED and read the pushbutton through serial communication (11.1, 11.2, 11.4)

Serial communication is used to send command and read values on an MCU board through a connected PC. For example, it is possible to send messages (like strings to be printed) to the PC and also receive commands. Use the non-blocking function to read and write on the UART, this is mandatory in case other tasks are performed by the MCU. You have to write a program that connects to the PC trough the UART connected to the USB cable and perform the following tasks:

a. Prints a welcome message and a simple menu.
b. Reads the data sent from the PC.
c. If '1' is passed, then toggles the LED.
d. If '2' is passed, then the status of the pushbutton is sent back (PRESSED | RELEASED).
e. If '3' is passed, then the menu is printed again.

The required steps are the following:

1. **Create a new project** using the "STM32CubeIDE".
2. **Configure the hardware** using "STM32Cube", selecting the proper configuration for the desired UART.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that prints a welcome message and the menu, and then wait for user commands.
5. **Write the correct ISR** for send and receive from the UART.
6. **Compile** the project.
7. **Download** the compiled code on the NUCLEO board.
8. **Use the provided python script to connect the PC to the NUCLEO board**.
9. **Test** the functionality of the circuit by sending different commands and looking at the board.

What happens if you send invalid commands? Try to manage that case.