– Elettronica dei Sistemi Digitali –
Lab#11

Direct Memory Access and Motor driving

The purpose of this laboratory session is to learn how to use the DMA. Furthermore, to learn how to drive a motor and read the rotation speed through a hall effect sensor. 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.

This laboratory assignment is optional! There will be no difference in the final evaluation.

Contents:

Abbreviations and acronyms:

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

1 - Sine wave generation (11.1) (OPTIONAL)

In this last exercise you will use the DMA. Read carefully the lab documentation before starting to program the board. Another application for the PWM is a low-cost digital-to-analog converter (DAC). By time varying the duty cycle percentage, a generic waveform can be generated. In this project, we want to generate a 50 Hz sine waveform with the NUCLEO board, by exploiting the PWM method. To achieve this goal, we will use the DMA feature together with the PWM. Firstly, we have to create the array of values for the PWM duty cycles. Then start the DMA: each time the period of the PWM is reached a new value is automatically loaded from memory and loaded in the duty cycle register. Read carefully the documentation to understand the principle behind this exercise. 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.
3. **Fully understand the code generated** by the “STM32Cube”.
4. **Write a C program** that evaluates the values for the PWM duty cycles and stores in a vector.
5. **Start the PWM in DMA mode.**
6. **Compile** the project.
7. **Build the correct circuitry** in order to filter the high frequencies.
8. **Download** the compiled code on the NUCLEO board.
9. **Test** the functionality using an oscilloscope.

Use another channel of the oscilloscope to plot the output of the board (not the output of the filter). What do you see? Why?

2 - Control a motor and reading rotation speed (11.2, 11.3, 11.4) (OPTIONAL)

We are getting to the end of this lab experiences, and you have now the knowledge to design a complete project. In this exercise you will use a real motor (LE149.12.43 from micromotors), and you will drive it using the NUCLEO board. That motor is a DC motor supplied at 12V with nominal speed of 60 RPMs, with a 43.3 ratio of reduction. The motor has an embedded hall effect sensor with all the condition circuitry integrated in a single onboard chip. The magnet rotating is configured to produce 3 pulse for rotation. It is possible to change the rotation speed of the motor changing the supply voltage. This is not done normally, instead a PWM signal can be used to perform this task. Since the MCU of our board is not able to drive directly our motor, you will use a L293 driver to translate the PWM signal of the NUCLEO to a 12V PWM signal. You have to design a software program for the MCU that change the rotation speed of the motor. The user sends a PWM duty cycle value to the board trough the serial line and receive the rotation speed in RPMs via the same interface. Use TIM3 to generate the PWM and TIM2 for reading the speed in input capture mode. Furthermore, print the speed twice every second: use the output compare on TIM2 to generate the delay. Carefully refer to the documentation for what concern the circuit you have to mount on the breadboard.

NB: you are going to connect different voltage domains to different components. Even if some protections are present on the components you are using, check twice before enabling the power supply. Remember to connect the freewheeling diode to prevent damage to the motor.

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, TIM2 and TIM3.
3. **Fully understand the code generated** by the “STM32Cube”.
4. **Write a C program** that, when a new value of duty cycle is received, updates the PWM of TIM3 and then wait for the next value.
5. **Write the ISR** for UART, input capture and output compare of TIM2.
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 values of the duty cycle.

What happens if you send invalid duty cycles? Check if the inserted value is out of the boundaries and send an error message to the user in that case. Use the oscilloscope to analyze both the PWM signal and the hall sensor output.

3 - Design an automatic closed loop control system for the motor (OPTIONAL)

In the previous exercise you designed an open loop control for the motor: setting the duty cycle resulted in changing the speed of the motor. Now you have to design a closed loop control system: reading from the user a desired speed and setting the duty cycle accordingly. Using the speed information coming from the hall effect sensor you can design a control network: firstly, use a correction factor proportional to the error, then use an integrative controller. You can find detailed information on how to design the control system in the documentation. You have to use TIM 3 to generate the PWM and also a temporization signal for your control function. Furthermore, use TIM2 for the same task as in the previous exercise. Set the initial speed and target speed to 0. 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, TIM2 and TIM3.

3. **Fully understand the code generated** by the “STM32Cube”.
4. **Write a C program** that, when a new value of speed is received, updates the corresponding variable and calls the controller when it is needed: remember that ISR should be as small as possible!
5. **Write the ISR** for UART, output compare on TIM3, input capture and output compare of TIM2.
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 values of the target speed.

What happens if you send invalid speed? Check if the inserted value is out of the boundaries and send an error message to the user in that case. Use the oscilloscope to analyze both the PWM signal and the hall sensor output. With the oscilloscope connected to the motor input set a target speed of 30RPM. Then change the voltage supply (6-18 V). What do you see on the oscilloscope? What happens to the speed of the motor?