POLITECNICO DI TORINO

– Elettronica dei Sistemi Digitali –
Lab#8

Square waveform generation and Potentiometer

The purpose of this laboratory session is to generate and measure a square waveform. Detailed information on how to program the board could be found on the LAB8 accompanying document [1] , and on the reference manual of the MCU. 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:

1 - Square waveform generator with TIM3: polling the counter register
2 - Square waveform generator with TIM3: polling UIF flag
3 - Square waveform generator with TIM3: using the Output Compare unit
4 - Square waveform generator with TIM3: using the Output Compare unit (automatic pin toggling)
5 - Square waveform generator with TIM3: frequency tuning with a potentiometer

Some of these projects require the availability of a few elementary measurement instruments, such as an oscilloscope and a waveform generator. For these projects, the LAB7 accompanying document describes an alternative option, 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. See the Appendix in LAB7 accompanying document for details on the related viewer application.

Abbreviations and acronyms:

MCU – Microcontroller Unit
TCR – Timer Count Register
ADC – Analog-to-Digital Converter

1 - Square waveform generator

There are several ways to implement a square wave generator with a microcontroller. In the following, we will analyze some possible solutions. The accuracy of the obtained waveform depends on the technique used to generate it. In

the Lab7 project, you used a software loop to generate the waveform. This is not a good idea, since normally MCUs are performing different tasks and the main loop could be interrupted during the execution of the software loop. A different approach is based on a specific peripheral, used to "count" time intervals by means of a free running counter. In this lab project, you will have to use one of the counters available in the STM32 MCU to generate square waves.

1.1 - Timer, Counter Register Polling

In the first project, we are going to use the TIM3 Counter to generate a 2 kHz square waveform on the PA10 pin. You have to poll the counter register and toggle the output pin only when the correct value is read. Detailed information on the counter configuration and usage can be found in [1] . You have to follow the list of steps given below. Do not start writing your code before step 4.

1. **Create a new project** using the STM32CubeIDE tool, then select the proper configuration for pin PA10 and counter TIM3.
2. **Fully understand the code generated** by STM32Cube.
3. **Follow the instruction** described in Section 2, at the end of this document.
4. **Write a C program** that toggles the output pin when the counter value is equal to a properly calculated target.
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? Try with different choices of the timer settings (particularly PSC and ARR registers)

What happens if you use different conditions when testing the content of the counter (e.g. "==" and ">=")? Do you see any difference on the oscilloscope?

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 the LAB7 accompanying document. The details of the viewer 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.

1.2 - Timer, Compare Flag Polling

You should have noticed that the repeated comparison between the content of the TIM3 counter and a defined constant may lead to a completely wrong behavior and this kind of approach must not be adopted in any similar application. In this project, you have to use the polling mode in the correct and safe way, which is based on the read of the UIF flag that the TIM3 peripheral sets at each programmed update event. Read carefully the text in [1] to understand the behavior of the timer and particularly the UIF. The required steps are:

1. **Create a new project** using the STM32CubeIDE, selecting the proper configuration for pin PA10 and counter TIM3.
2. **Fully understand the code generated** by the STM32Cube.
3. **Follow the instruction** described at the end of this document, in Section 2
4. **Write a C program** that toggles the output pin when the UIF flag of TIM3 is set by the counter.
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 code to achieve a frequency of 4 MHz, without creating a new project.

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 the LAB7 accompanying document. The details of the viewer 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.
As for the last point, the generation of a 4 MHz waveform, this frequency is too high for the viewer capabilities and the generated signal could not be sampled. Therefore, just skip this point if you do not have an oscilloscope available.

1.3 - Timer Output Compare Function

The approach adopted in the previous project (the one based on the UIF polling) is not always available. Some (old) MCUs have timers where the user is not able to configure the update event. Furthermore, in some cases you need to generate multiple waveforms, and this cannot be done by directly using the UIF flag. On the contrary, multiple waveforms can be generated easily by means of the Output Compare (OC) unit. Each timer in the STM32 has multiple channels that can be configured separately. Thus, by exploiting the OC unit, we can set different configuration options for each channel. In this project, you have to use the OC function to generate two square waves: a 2.5 kHz signal on PA10 pin, using channel 1 of TIM3 and a 12.5 kHz wave on PB10 pin, using channel 2 of the same timer. The two corresponding flags must be used in polling mode to update both the output pins and the OC target values. The required steps are:

1. **Create a new project** by using STM32CubeIDE, select the proper configuration for the involved pins and the TIM3 unit. To enter the proper configuration, follow the suggestions given in [1] .
2. **Fully understand the code generated** by STM32Cube.
3. **Follow the instruction** described at the end of this document (Section 2).
4. **Write a C program** that exploits the OC to generate the square wave.
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?

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 the LAB7 accompanying document. The details of the viewer 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. Moreover, the frequency of the two square waveforms must be rescaled by a factor 10, i.e. 250 Hz for the signal on the PA10 pin, and 1.25 kHz for the PB10 pin.

1.4 - Timer Output Compare Function, with automatic pin toggling

The OC Function of the timer peripheral has another useful property. You can set it in order to automatically toggle a specific pin in hardware when the match occurs. The only thing that you have to manage is the update of the target value for the OC. In this project, you have to exploit the OC Toggle on match function to generate a 2.5 kHz signal on PA6 pin, using channel 1 of TIM3, and a 12.5 kHz wave on PA7 pin, using channel 2 of the same timer. The two corresponding flags must be used in polling mode to update the OC target values. The required steps are:

1. **Create a new project** by using STM32CubeIDE, then select the proper configuration for the involved pins and the TIM3 unit. To enter the proper configuration, follow the suggestions given in [1] .
2. **Fully understand the code generated** by STM32Cube.
3. **Follow the instruction** described at the end of this document (Section 2).
4. **Write a C program** that exploits the OC to generate the square wave.
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?

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 the LAB7 accompanying document. The details of the viewer 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. Also for this project, scale down the signal frequency by a factor 10.

1.5 - Timer Output Compare Function with Variable frequency

The objective of this project is changing the frequency of the generated square wave by means of a potentiometer. The potentiometer is connected to a GPIO pin configured as an analog input and the internal analog to digital converter (ADC) is used to sample and convert the input signal into a digital value. Finally, the obtained digital sample is exploited to tune the frequency in the interval 800 Hz to 4.5 kHz. We acquire samples from the ADC using the polling approach, which means that we check the end of conversion flag to know when the conversion is complete. On the other hand, to generate the square wave, we exploit the TIM3 timer configured as in the previous project (OC in automatic pin toggling mode). The required steps are the following:

1. **Create a new project** using STM32CubeIDE and enter the proper configuration for the output pin, the TIM3 unit and the ADC [1] .
2. **Fully understand the code generated** by the STM32Cube.
3. **Follow the instruction** described at the end of this document (Section 2).
4. **Write a C program** that polls the flag of the end of conversion of the ADC and updates the correct register of the timer. You may need to perform mathematical operations on the converted value to reach the desired interval of frequencies.
5. **Compile** the project.
6. **Download** the compiled code on the Nucleo board.
7. Use wires, a breadboard and a resistor to connect a discrete potentiometer to analog input of the Nucleo board. You have to use the ground and the supply voltage available on the NUCLEO board. See image in Figure *1*.
8. **Test** the functionality of the circuit by using the oscilloscope and varying the potentiometer.
9. **Use the debugger** to report the value of the OC register for the following frequencies:

Frequency	OC Register
800 Hz	
1.2 kHz	
3.0 kHz	
4.5 kHz	

Is the waveform stable? Do you see any strange behavior on the oscilloscope?

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 the LAB7 accompanying document. The details of the viewer 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. In addition, scale down the signal frequency by a factor 10 and use the 80 – 450 Hz range.
Finally, as the viewer application heavily uses the ADC, it is not recommended to connect the potentiometer to the ADC input to control the frequency of the waveform. Therefore, you have to exploit the user pushbutton to determine the frequency. Initially, just after launching the application, the generated waveform must have the lowest frequency (80 Hz); then, when you press the pushbutton, the frequency must grow to the next level in the table above (120 Hz) and at any further pushing of the button the frequency must update the waveform with the next frequency value in the table, up to 450 Hz. After reaching this frequency level, the next frequency update action must lead back to the 80 Hz.

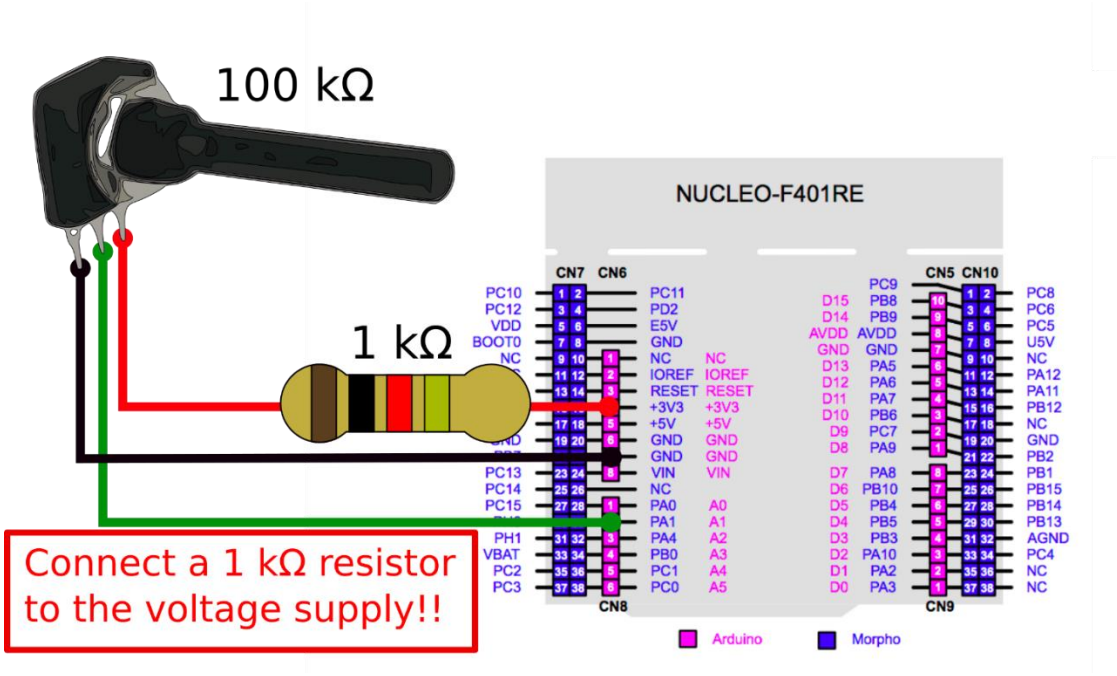

Figure 1: Connecting a potentiometer to the Nucleo board.

2. Additional code to be added to assigned projects

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=0x12c0; // What is this number ?
for(int i=0;i<x;i++);
  x = (x >> 2) | (((x & 1)^(x & 2)) << 4);
```

References

[1] Lab8-STM32-document, Polling timers and ADC