Politecnico di Torino

– Elettronica dei Sistemi Digitali –
Lab#9

Interrupts

The purpose of this laboratory session is to learn how to manage several tasks on the NUCLEO processors by the means of interrupts. 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 have to follow all the assignments. If some questions are present at the end of the assignments, be ready to answer in the lab report.

Contents:

1. Using the Timer Count Register in interrupt mode
2. Managing several interrupts

Abbreviations and acronyms:

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

1- Interrupt-based variable frequency square waveform generator (9.1, 9.2, 9.3)

Polling is not always the best way to make things. Interrupts, differently from polling, do not occupy the processor in waiting loops, but *interrupt* the execution when an event occurs. In this exercise you have to implement the same functions of the last exercise of the previous lab using an interrupt to toggle the output pin. Do not use *Toggle on match* functionality, instead toggle PA10 in the Interrupt Service Routine (ISR). In this case, you have to follow the pass described at the end of this assignment before starting to write your program. 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 PA10, counter TIM3 and the ADC. Set preemptive priority for TIM3 interrupt. **Configure the source clock of TIM3 to 84MHz. Fully understand the code generated** by the "STM32Cube".
3. **Follow the instruction** described at the end of this assignment.
4. **Write a C program** that polls the flag of the end of conversion of the ADC and updates the correct variable. You may need to perform mathematical operation on the converted value to reach the desired interval of frequencies. The TIM3 is configured in OC, but we use the interrupt.
5. **Write the Interrupt Service Routine (ISR)** for TIM3.
6. **Compile** the project.
7. **Download** the compiled code on the Nucleo board.
8. Use a breadboard and the discrete potentiometer to **build the necessaire circuitry**. You have to use the NUCLEO board to give the supply voltage to the potentiometer.
9. **Test** the functionality of the circuit by using the oscilloscope and varying the potentiometer.

Measure the minimum and maximum period/frequency of the waveform, are the same of last lab? Are the measures affected by jitter?
Now we discover that the template provided for the last lab already contained the management of an interrupt. This interrupt was intended to emulate the real behavior of an MCU where the execution of a task can be interrupted by another task.
Looking at the code you will discover that the SysTick was used to this purpose. In particular, there was a routine to simulate some tasks called periodically. Even if you know the purpose, insert this function as in the last lab.

Even if you know the purpose, insert this function as in the last lab.

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)"

```
int x = (rand()%10 + 50)*5;
   for(int i=0;i<x;i++);
```

Now try to remove the pre-emptive priority for the TIM3 interrupt. Do you see the same behavior?

2- Multiple interrupts

In the previous exercise you were asked to write the code to manage just one interrupt. In the following you will run several tasks on the MCU. Each task will be managed as an interrupt event through a routine.

2.1 - Three interrupts (9.1, 9.2, 9.3)

Prepare a project and develop the C code to implement a 3 bit clock process working in interrupt. The 3 bit clock process relies on three channels of the Timer Count Register of TIM3, namely channel 1, channel 2, channel 3. Exploit the OC mode that automatically toggles a hardwired pin of the MCU. The interrupts produced by the output compare channels are used as follows:

1. channel 1 (connected to an External LED) has to produce a square waveform with period T_0=1 ms.
2. channel 2 (connected to an External LED) has to produce a square waveform with period T_1=2 ms.
3. channel 3 (connected to an External LED) has to produce a square waveform with period T_2=4 ms.

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 channel 1,2 and 3 of TIM3.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that starts the counter and then enter an infinite loop.
5. **Write the Interrupt Service Routines (ISR)** for all the channels of TIM3.
6. **Compile** the project.
7. **Download** the compiled code on the Nucleo board.
8. **Test** the functionality of the circuit.

Are you able to see all the waveform stable on the oscilloscope? Why?

2.2- Four interrupts (9.4)

Modify the previous project to add the following task: toggle built-in LED status on PA5 (from on to off and vice versa) every time the pushbutton is pressed. Implement this feature as an interrupt. 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 channel 1,2 and 3 of TIM3, the built-in LED and the pushbutton.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that starts the counter and then enter an infinite loop.
5. **Write the Interrupt Service Routine (ISR)** for all the channels of TIM3 and the pushbutton.
6. **Compile** the project.
7. **Download** the compiled code on the Nucleo board.
8. **Test** the functionality of the circuit by using the oscilloscope and pressing the pushbutton.

Is the pushbutton afflicting the square waves frequency? Does it insert jitter? Why?

2.3- Five interrupts

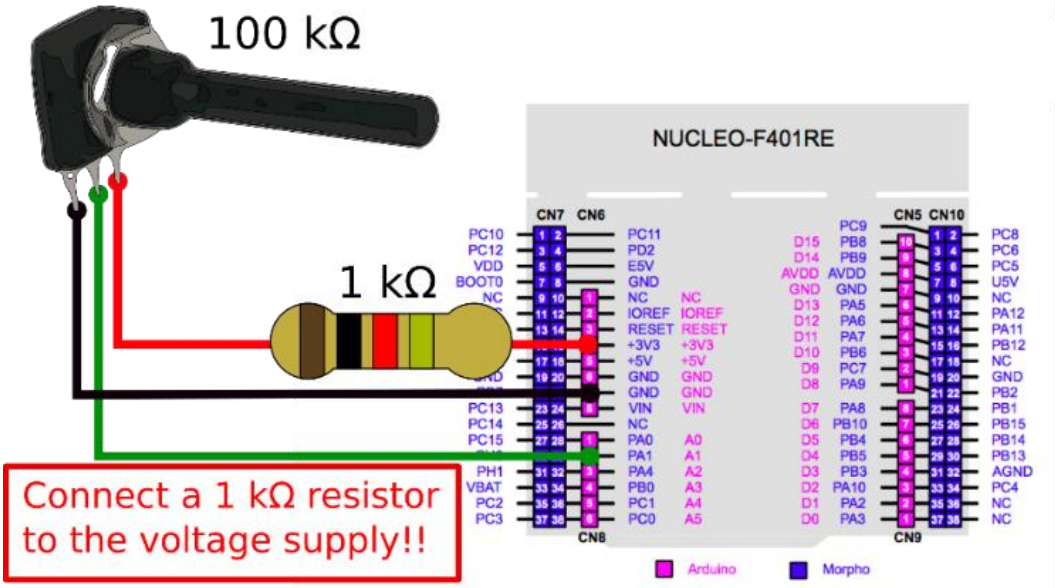
Modify the previous project to add the following task: use the potentiometer to change the period of the three square waves. The range should be:

1. channel 1 → 0.1-1 ms
2. channel 2 → 0.2-2 ms.
3. channel 3 → 0.4-4 ms.

Furthermore, use channel 1 of the ADC as in the previous experience but in single conversion mode. You have to use channel 2 of TIM4 in OC to trigger a new conversion of the ADC each 500 ms. The TIM4 must be configured in interrupt, and you have to start a new conversion each time the CNT value is equal to the CCR2. Use the polling approach for reading the new converted value and updating the frequencies. 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 channel 1,2 and 3 of TIM3, channel 2 of TIM4, the LED, the pushbutton and the ADC.
3. **Fully understand the code generated** by the "STM32Cube".
4. **Write a C program** that starts the counters and then polls the ADC status (**in the main function**).
5. **Write the interrupt service routines (ISR)** for all the channels of TIM3, the pushbutton and channel 2 of TIM4.
6. **Compile** the project.
7. **Download** the compiled code on the Nucleo board.
8. **Test** the functionality of the circuit by using the oscilloscope, pressing the pushbutton and changing the value of the potentiometer.

Are you able to see all the waveform stable on the oscilloscope? Why? Do you see a different behavior when you change the potentiometer? Why?


100 kΩ
1 kΩ
NUCLEO-F401RE
CN7 CN6
CN5 CN10
CN8
CN9
Arduino
Morpho
Connect a 1 kΩ resistor
to the voltage supply!!