**Session 4**

**-**

**Interrupts, timers**

**Author:** Guillermo Cortés Orellana

**Teacher:** Tomáš Frýza

****

Lab assignment

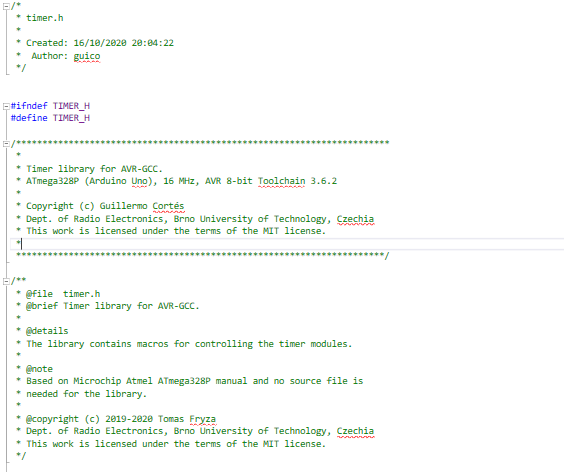
1. **Preparation tasks**

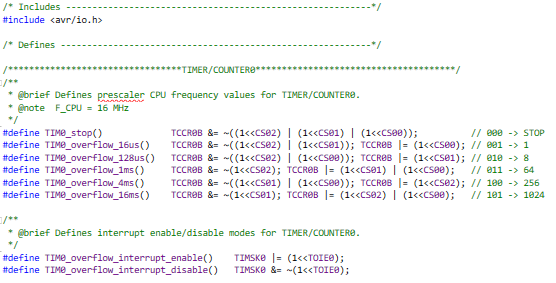
* Table with overflow times

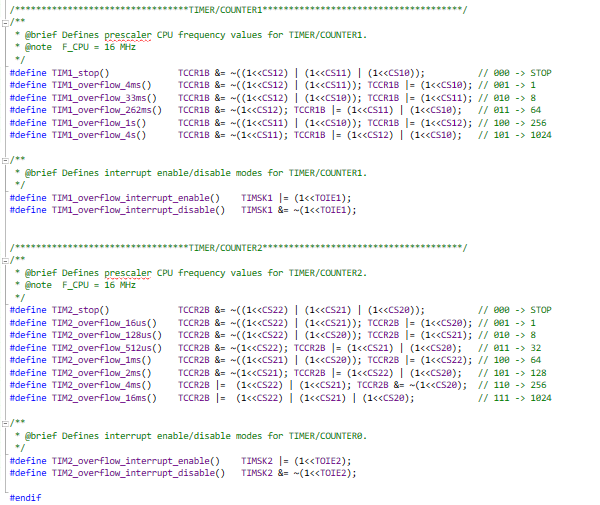
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Module | Number of bits | 1 | 8 | 32 | 64 | 128 | 256 | 1024 |
| Timer/Counter0 | 8 | 16u | 128u | - | 1024u | - | 4096u | 16384u |
| Timer/Counter1 | 16 | 4096 | 32768u | - | 262144u | - | 1’04 | 4’19 |
| Timer/Counter2 | 8 | 16u | 128u | - | 1024u | - | 4096u | 16384u |

1. **Timer library**

* Listing of library header file ***timer.h***

****

****



You can find the code on my GitHub:

https://github.com/GuicoRM/Digital-Electronics-2

* Table with ATmega328P selected interrupts sources

|  |  |  |  |
| --- | --- | --- | --- |
| Program address | Source | Vector name | Description |
| 0x0000 | RESET | - | Reset of the system |
| 0x0002 | INT0 | INT0\_vect | External interrupt request number 0 |
| 0x0004 | INT1 | INT1\_vect | External interrupt request number 1 |
| 0x0006 | PCINT0 | PCINT0\_vect | Pin change interrupt request 0 |
| 0x0008 | PCIN1 | PCINT1\_vect | Pin change interrupt request 1 |
| 0x000A | PCINT2 | PCINT2\_vect | Pin change interrupt request 2 |
| 0x000c | WDT | WDT\_vect | Watchdog time-out interrupt |
| 0x0012 | TIMER2\_OVF | TIMER2\_OVF\_vect | Overflow of Timer/Counter2 value |
| 0x0018 | TIMER1\_COMPB | TIMER1\_COMPB\_vect | Compare match between Timer/Counter1 value and cannel B compare value |
| 0x001A | TIMER1\_OVF | TIMER1\_OVF\_vect | Overflow of Timer/Counter1 value |
| 0x0020 | TIMER0\_OVF | TIMER0\_OVF\_vect | Overflow of Timer/Counter0 value |
| 0x0024 | USART\_RX | USART\_RX\_vect | USART RX complete |
| 0x002A | ADC | ADC\_vect | ADC conversion complete |
| 0x0030 | TWI | TWI\_vect | 2-wire serial interface |

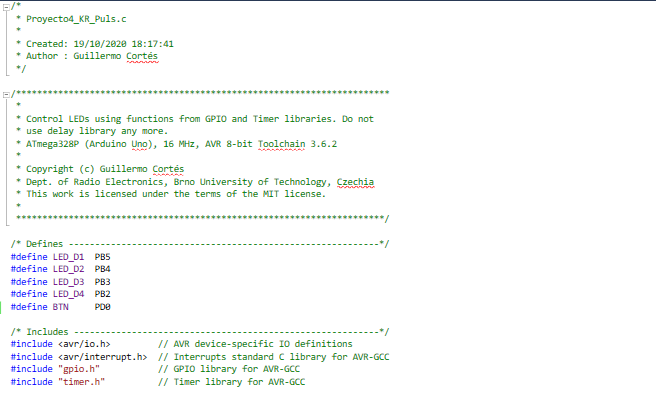
* Listing of the final application ***main.c***

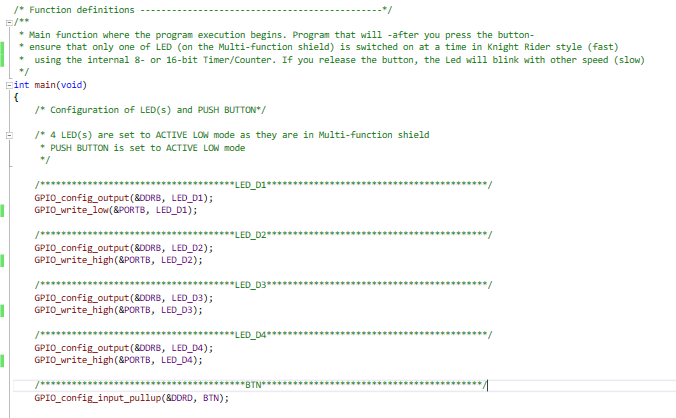
I didn’t get LEDS blink in Knight-Rider style at a certain speed using ISR when I push the push button and at another speed when I don’t push it. I would swear the code it’s correct, but I don’t know why it doesn’ work.

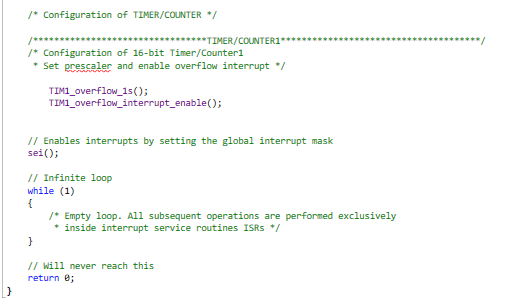
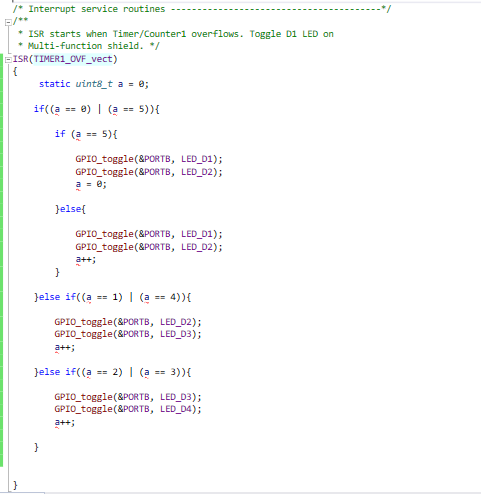
Nevertheless, I got LEDS blink in Knight-Rider style using ISR but WITHOUT push Button and only with one kind of speed.

I attach both cases:

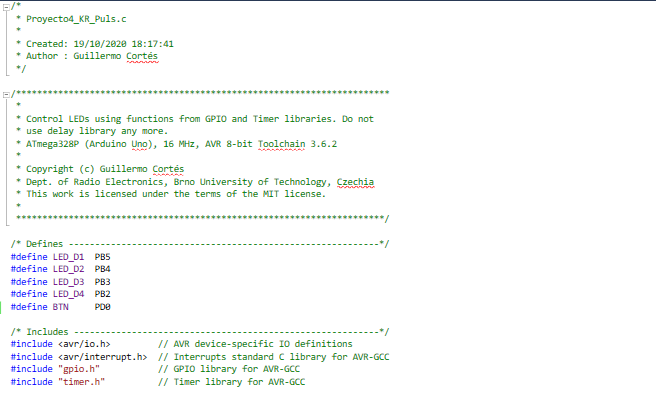
* **Case 1**: LEDS blink in Knight-Rider style (one kind of speed) using ISR but WITHOUT push button:

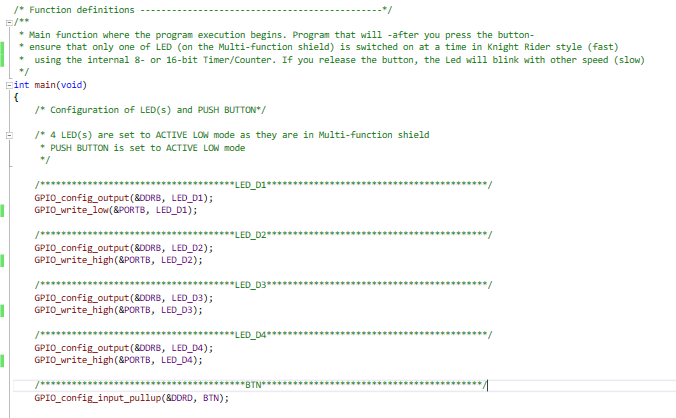


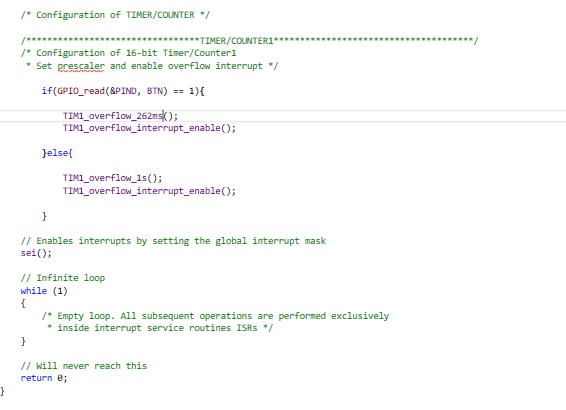
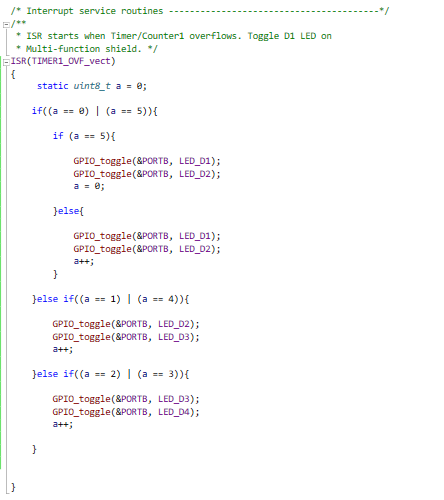




* **Case 2**: LEDS don’t blink in Knight-Rider at different speeds style using ISR and WITH push button:





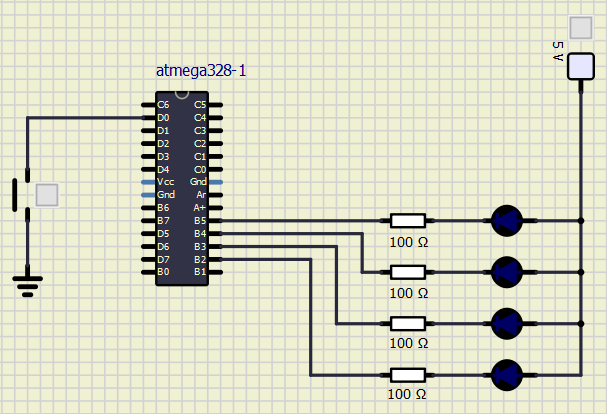


You can find the code on my GitHub:

https://github.com/GuicoRM/Digital-Electronics-2

* Screenshot of SimulIDE circuit

**Note:** LEDS are ACTIVE LOW as ‘Multi-function shield’

****

* In your Word, describe the difference between a common C function and interrupt service routine
* **C functions** are blocks of code (tools) designed by the programmer for a specific purpose. They are usually used in different parts of the code at a certain point in time. Meanwhile, **ISR (I**nterrupt **S**ervice **R**outine**)** are blocks of code that run asynchronously and therefore somehow interrupt the work of the preocessor.

1. **PWM**

* Table with PWM channels of ATmega328P

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Description | MCU pin | Arduino pin |
| Timer/Counter0 | OC0A | PD6 | 6 |
| Timer/Counter0 | OC0B | PD5 | 5 |
| Timer/Counter1 | OC1A | PB1 | 9 |
| Timer/Counter1 | OC1B | PB2 | 10 |
| Timer/Counter2 | OC2A | PB3 | 11 |
| Timer/Counter2 | OC2B | PD3 | 3 |

* Describe the behavior of Clear Timer on Compare and fast PWM modes

In Clear **Timer on Compare** or CTC mode (WGM02:0 = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

On the other hand, **Fast Pulse Width Modulation or fast PWM** mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option.