Jacobs University Bremen

Natural Science Laboratory

Embedded Systems Laboratory
CO26-300312

Fall Semester 2017

External Interrupt : Assembler

Mailbox :
104

Instructor :
Ph.D. Fangning Hu

Author :
Biruk Amare

26 septembre 2017

JACOBS
UNIVERSITY

1 Introducion

The main objective of the lab is to design a circuit showing that when pressing a button, an external interrupt should be triggered using ATmega328. Then, study how voltage changes on the pins `INT0` or `INT1` will trigger the corresponding external interrupt. The other purpose of the lab is to understand how to set the `EICRA` and `EMISK` to enable external interrupt.

Interrupts are essentially events that require immediate attention by the microcontroller. When an interrupt event occurs the microcontroller pauses its current task and attends to the interrupt by executing an `Interrupt Service Routine (ISR)`. At the end of the `ISR`, the microcontroller returns to the task it had paused and continue its normal operations.

In order for the microcontroller to respond to an interrupt event feature of the microcontroller must be enabled along with the specific interrupt. This is done by setting the *Global Interrupt Enabled* bit and the *Interrupt Enable* bit of the specific interrupt.

1.1 Interrupt Service Handler

An *Interrupt Service Routine (ISR)* or *Interrupt Handler* is a piece of code that should be executed when an interrupt is triggered. Usually, each enabled interrupt has its own ISR. In AVR assembly language each ISR must end with the RETI instruction which indicates the end of the ISR.

1.2 Interrupt Flags and Enabled bits

Each interrupt is associated with two bits, an `Interrupt Flag Bit` and an `Interrupt Enabled Bit`. These bits are located in the I/O registers associated with the specific interrupt :

— The interrupt flag bit is set whenever the interrupt event occurs, whether or not the interrupt is enabled. The interrupt enabled bit is used to enable or disable a specific interrupt. Generally, it tells the microcontroller whether or not it should respond to the interrupt if it is triggered. So, basically both the Interrupt Flag and the Interrupt Enabled are required for an interrupt request to be generated as shown in the figure below.

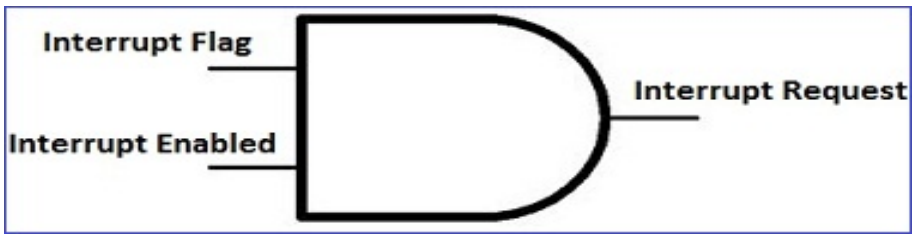

Figure 1 – Interrupt Bits

1.3 Global Interrupt Enabled Bit

Apart from the enabled bits for the specific interrupts the global interrupt enabled bit **must** be enabled for interrupts to be activated in the microcontroller. For the AVR 8-bits microcontroller this bit is located in the `Status I/O Register (SREG)`. The *Global Interrupt Enabled is bit 7*, the I bit, in the SREG.

A wire from the button is connected to pin7 on the arduino board. A $10k\Omega$ pull-up resistor as shown on the circuit diagram where the input voltage to the pin keeps high before pressing and becomes low after pressing the button is connected as well.

2 Prelab

2.1 Read the ATmega328 datasheet (Chapter External Interrupts) and find out the meaning of each pin in the necessary registers. You will need to set the corresponding bit in register EMISK to enable the external interrupt and set the corresponding bits in EICRA to control which types of voltage change will trigger the interrupt. Find the I/O addresses for registers EMISK and EICRA. Note that the command OUT can only access the lower 64 I/O registers.

This is discussed in the Introduction part of this lab report.

2.2 Read the ATmega328 datasheet (Chapter Interrupts) and find out the program address for the External Interrupt Vector in the Interrupt Vector Table. Be careful to the different devices in the datasheet, they have different Tables!!

This is also discussed in the Introduction part of this lab report.

3 Lab Assignments

3.1 Check the Ardunino Schematic to find out which pin you should use to connect to the button. Design your circuit such that when pressing the button, an external interrupt will be triggered. A typical way to do that is using a 10K Ohm pull-up resistor as shown in the circuit diagram where the input voltage to the pin keeps high before pressing and becomes low after pressing the button.

The schematics used for this task are shown below :

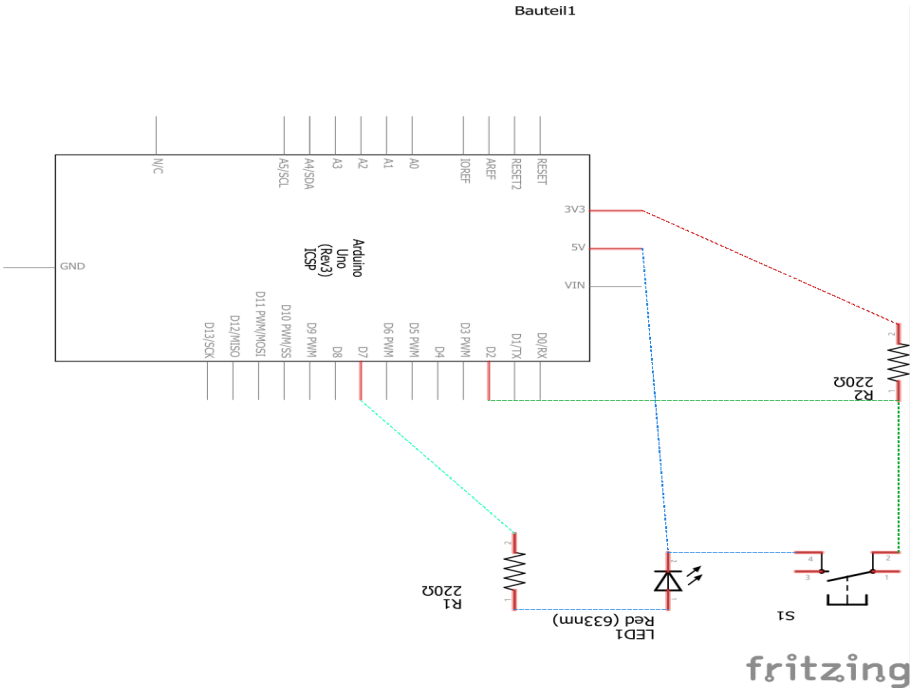

FIGURE 2 – Schematic plan 1

The corresponding schematic in setup view :

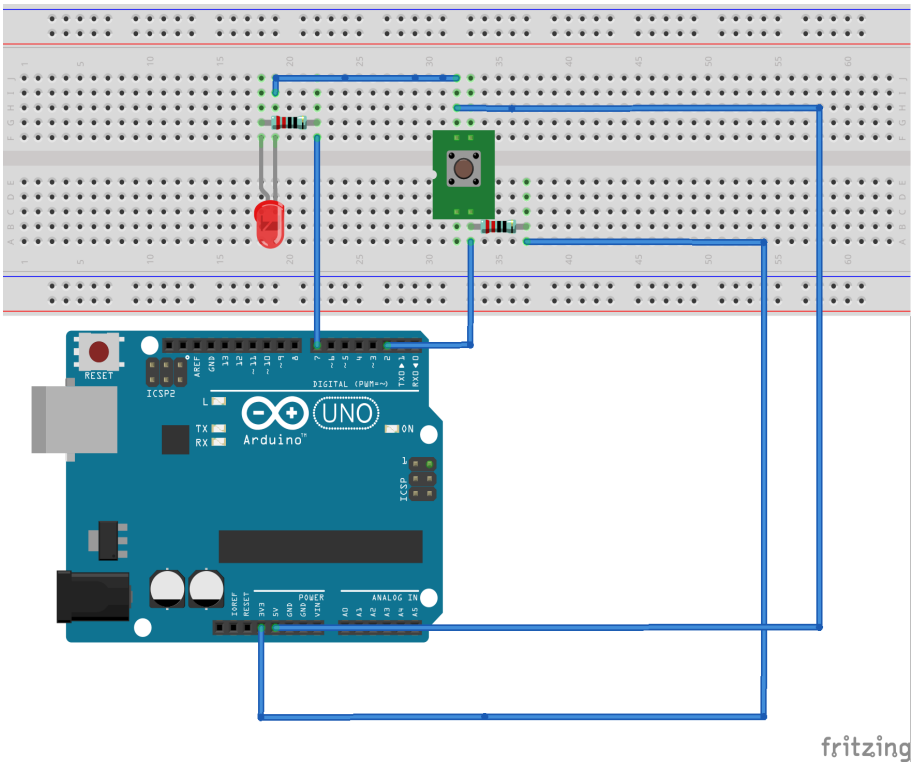

FIGURE 3 – Schematic Plan 2

3.2 Design what you want the LED does when pressing the button. In my case, I make it quickly blink twice when pressing the button. Otherwise it blink slowly in the normal situation. Write your Interrupt Service Routine codes to finish your design.

```
;
; Lab2.asm
;
; Created: 09.09.2017 15:04:08
; Author : biruk
;

.include "m328def.inc"
.org 0x0000
JMP begin ; Jump to begin
JMP ext_int0
JMP ext_int1
.org 0x0034

begin:
        ; Initialize the microcontroller stack pointer
        LDI R16, low(RAMEND)
        OUT SPL, R16
        LDI R16, high(RAMEND)
        OUT SPH, R16

        ; Set external interrupt control and mask register
        LDI R16, 0b00001010
        STS EICRA, R16
        LDI R16, 0b00000011
        OUT EIMSK, R16
        LDI R16, 0b00000001 ; PD7 as ouput
        OUT DDRD, R16
        LDI R16, 0xFF
        OUT PORTD, R16 ; Enabling internal pull-up
        SEI
mainloop:
        NOP ; No operation (wait do nothing)
        NOP
        NOP
        RJMP mainloop

ext_int0:
        LDI R17, 0b11111111
```

```
        IN R16, PORTD ; read PortD
        LDI R18, 0b11111110
        CPSE R16, R18 ; compare R16 and R18 and skip if equal
        LDI R17, 0b11111110
        OUT PORTD, R17
        RETI
ext_int1:
        NOP
RETI
```

4 References

[1] http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-
datasheet_Complete.pdf1

[2] http://embsys-fhu.user.jacobs-university.de/?page_id=49