Linnéuniversitetet
Kalmar Växjö

Lab Report

Lab 3

Arduino UNO, REV 3 – ATmega 328p

Author(s): David Mozart, Marcus Thornemo Larsson
Supervisor: Cristian Babes
Semester: HT19
Course Code: 1DT301

Table of Content

Task I

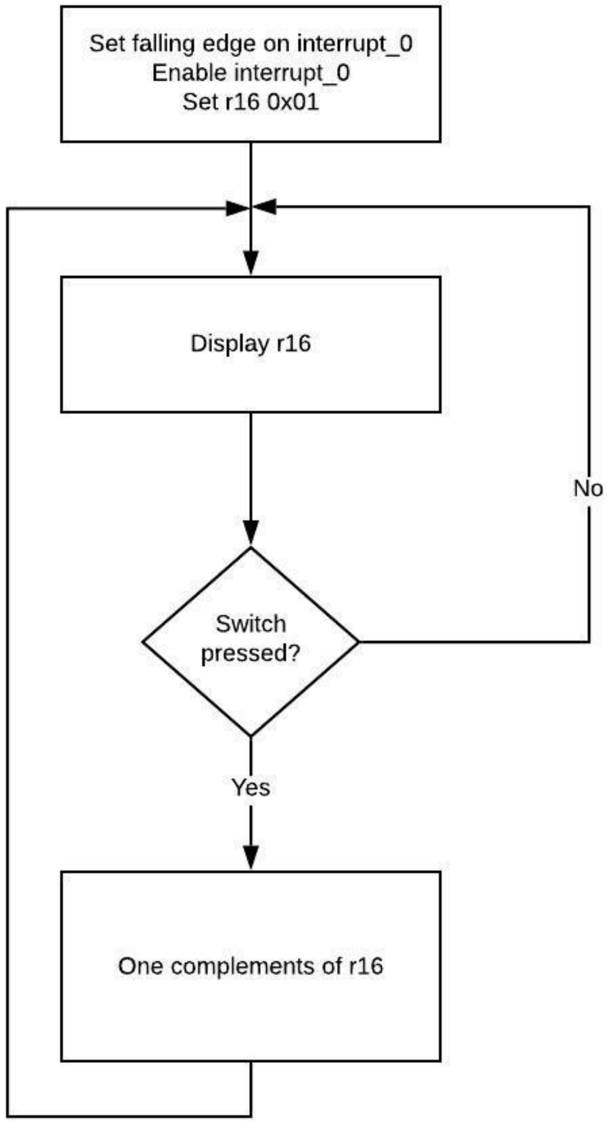
Description

This program toggles a LED using interrupts, more specifically interrupt0. Each time the button is pressed, the LED goes ON if it was previously OFF and OFF if it was on before the switch press. The interrupt routine toggles the r16 register (By taking its one complement each time the switch is pressed) and out its content to the LED.

Assembly

```
.include "m328pdef.inc"
.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0

.org 0x72

start:
    ldi r20, HIGH(RAMEND)
    OUT SPH, R20
    ldi R20, low(RAMEND)
    out SPL, R20

    ldi r16, 0x01
    out DDRB, r16

    ldi r16, 0b0000_0010
    sts EICRA, r16

    ldi r16, 0b0000_0001
    out EIMSK, r16
sei

ldi r16, 0x01

main_program:
    out PORTB, r16
rjmp main_program

interrupt_0:
    com r16
reti
```

Flowchart


Set falling edge on interrupt_0
Enable interrupt_0
Set r16 0x01
Display r16
Switch pressed?
No
Yes
One complements of r16

Task II

Description

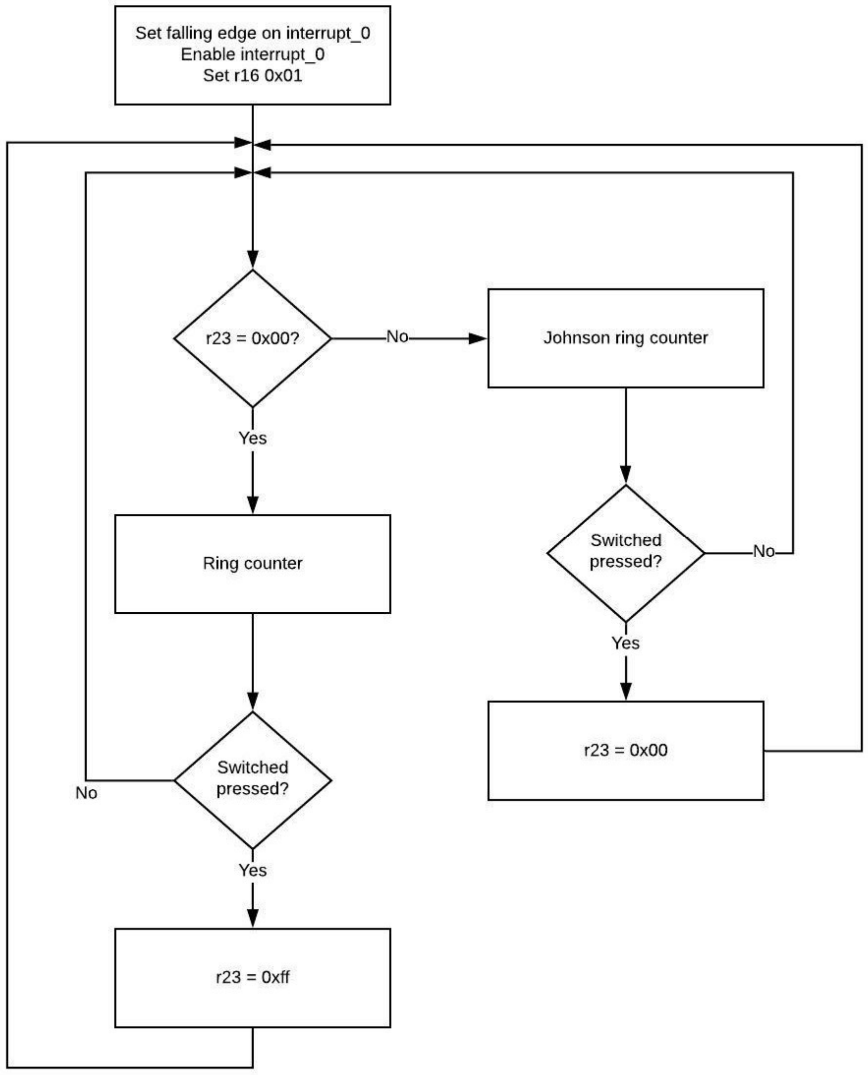
This program switches between Johnson counter and ring counter whenever the switch is pressed using interrupts.

Assembly

```
.include "m328pdef.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0

.org 0x72

start:
    ldi r20, HIGH(RAMEND)
    out SPH, R20
    ldi R20, low(RAMEND)
    out SPL, R20

    ldi r16, 0xff
    out DDRB, r16

    ldi r16, 0b0000_0010
    sts EICRA, r16

    ldi r16, 0b0000_0001
    out EIMSK, r16

    ldi r16, 0b0000_0001
    mov r17, r16
    ldi r23, 0x00
sei

ring_counter:
    cpi r23, 0xff
    breq johnson_counter_inc
    out PORTB, r16
    rcall delay
    cpi r23, 0xff
    breq johnson_counter_inc
    cpi r16, 0b0010_0000
    breq reset_ring_counter
    lsl r16
jmp ring_counter
```

```
reset_ring_counter:
    ldi r16, 0b0000_0001
jmp ring_counter


johnson_counter_inc:
    out PORTB, r16
    rcall delay
    cpi r23, 0x00
    breq ring_counter
    add r16, r17
    cpi r17, 0b0100_0000
    breq johnson_counter_dec
    lsl r17
jmp johnson_counter_inc

johnson_counter_dec:
    lsr r17
    sub r16, r17
    out PORTB, r16
    cpi r17, 0b0000_0001
    breq johnson_counter_inc
    rcall delay
    cpi r23, 0x00
    breq ring_counter
jmp johnson_counter_dec


delay:
    ldi  r18, 41
    ldi  r19, 150
    ldi  r20, 128
L1:
    dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
reti

interrupt_0:
    com r23
    ldi r16, 0b0000_0001
    ldi r17, 0b0000_0010
reti
```

Flowchart


Set falling edge on interrupt_0
Enable interrupt_0
Set r16 0x01
r23 = 0x00?
No
Johnson ring counter
Yes
Ring counter
Switched pressed?
No
Yes
r23 = 0x00
Switched pressed?
No
Yes
r23 = 0xff

Task III

Description

This program illustrates the 6 rear lights of a car, three to the left, and three to the right. When not turning, the lights are constantly lit (lights 1, 2, 5 and 6). Once the left switch is pressed, lights 1 and 2 will turn on, and lights 4, 5 and 6 will display a ring counter. Similarly, if the right switch is pressed, lights 5 and 6 will turn on and lights 1, 2 and 3 will display a ring counter. Pressing the same switch, a second time will put the lights in a non-turning state. It is also possible to press the right button while turning left, that puts the program in a right turn state.

Assembly

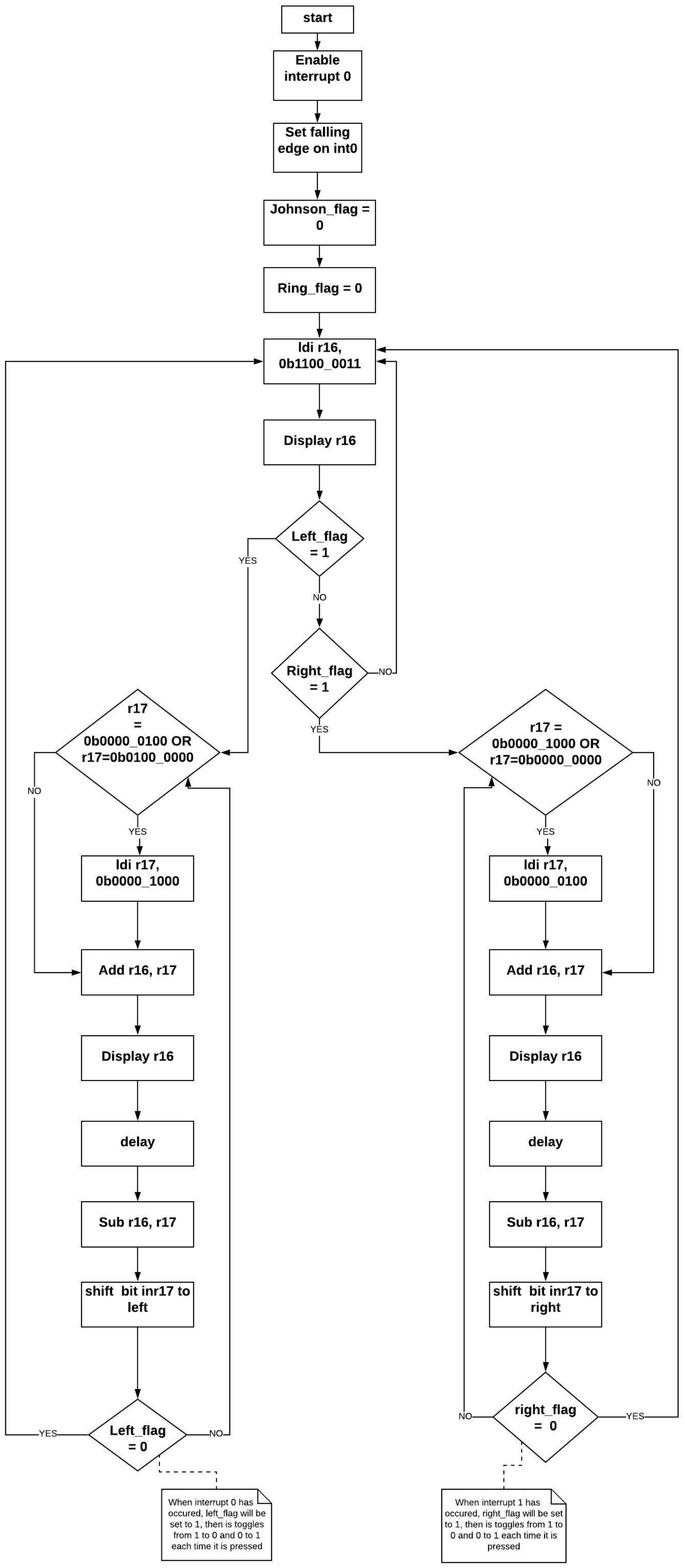
```
.include "m328pdef.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_right

.org INT1addr
rjmp interrupt_left

.org 0x72

start:
    ldi r20, HIGH(RAMEND)
    out SPH, R20
    ldi R20, low(RAMEND)
    out SPL, R20

    ldi r16, 0xff
    out DDRB, r16
    clr r16
    out PORTB, r16

    ldi r16, 0b0000_1010
    sts EICRA, r16

    ldi r16, 0b0000_0011
    out EIMSK, r16

    ldi r16, 0b0011_0011
    ldi r22, 0x00
    ldi r23, 0x00
sei
```

```
main:
    ldi r16, 0b0011_0011
    out PORTB, r16
    cpi r22, 0xff
    breq turning_left
    cpi r23, 0xff
    breq turning_right
rjmp main

turning_left:
    cpi r17, 0b0000_0100
    breq reset_left
    ldi r16, 0b0000_0011
    add r16, r17
    out PORTB, r16
    rcall delay
    sub r16, r17
    lsl r17
    cpi r17, 0b0100_0000
    breq reset_left
    cpi r22, 0x00
    breq main
rjmp turning_left

reset_left:
    ldi r17, 0b0000_1000
rjmp turning_left

turning_right:
    cpi r17, 0b0000_1000
    breq reset_right
    ldi r16, 0b0011_0000
    add r16, r17
    out PORTB, r16
    rcall delay
    sub r16, r17
    lsr r17
    cpi r17, 0b0000_0000
    breq reset_right
    cpi r23, 0x00
    breq main
rjmp turning_right

reset_right:
    ldi r17, 0b0000_0100
rjmp turning_right

interrupt_left:
    com r22
    clr r23
    ldi r16, 0b0000_1011
    ldi r17, 0b0000_1000
reti
```

```
interrupt_right:
    com r23
    clr r22
    ldi r16, 0b0011_0100
    ldi r17, 0b0000_0100
reti

delay:

    ldi  r18, 41
    ldi  r19, 150
    ldi  r20, 128
L1:
            dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
reti
```

Flowchart

See next page...


start
Enable interrupt 0
Set falling edge on int0
Johnson_flag = 0
Ring_flag = 0
ldi r16, 0b1100_0011
Display r16
Left_flag = 1
YES
NO
Right_flag = 1
NO
YES
r17 = 0b0000_0100 OR r17=0b0100_0000
NO
YES
ldi r17, 0b0000_1000
Add r16, r17
Display r16
delay
Sub r16, r17
shift bit inr17 to left
Left_flag = 0
YES
NO
When interrupt 0 has occured, left_flag will be set to 1, then is toggles from 1 to 0 and 0 to 1 each time it is pressed
r17 = 0b0000_1000 OR r17=0b0000_0000
NO
YES
ldi r17, 0b0000_0100
Add r16, r17
Display r16
delay
Sub r16, r17
shift bit inr17 to right
right_flag = 0
NO
YES
When interrupt 1 has occured, right_flag will be set to 1, then is toggles from 1 to 0 and 0 to 1 each time it is pressed