Lab 3 Report

Authors: Andrei Neagu
Konstantinos Tatsis
Course code: 1DT301

Contents

1 Task 1 **1**
1.1 Flowchart 1 2

2 Task 2 **3**
2.1 Flowchart 2 5

3 Task 3 **6**
3.1 Flowchart 3 8

4 Task 4 **9**
4.1 Flowchart 4 12

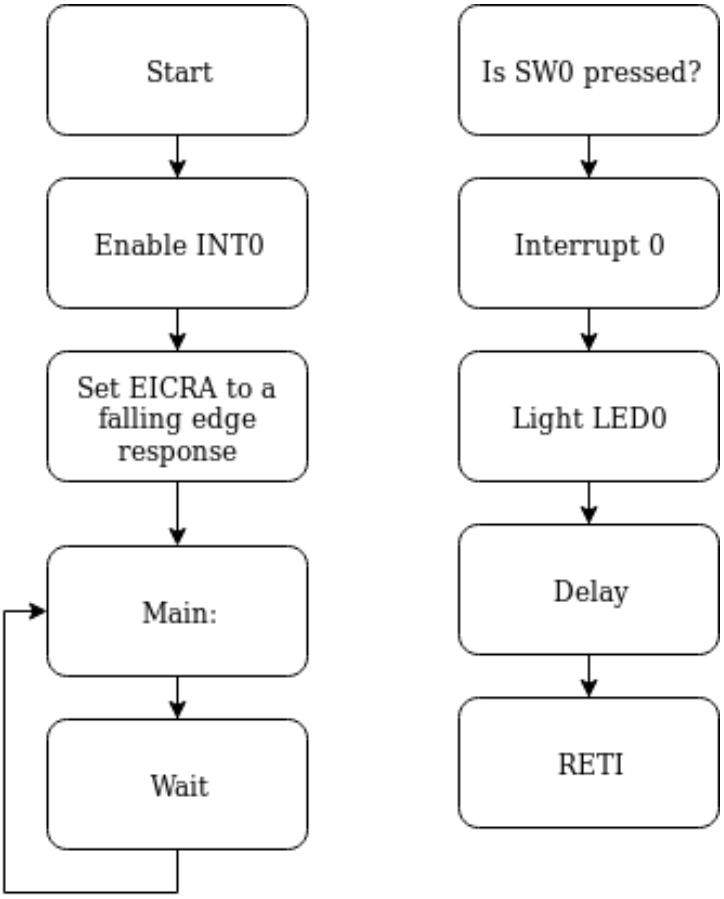
1 Task 1

Write a program that turns ON and OFF a LED with a push button. The LED will be extinguished when pressing the button. The program will use Interrupt. Connect the push buttons to PORT D. The program should have a main program that runs in a loop and wait for the interrupts. An interrupt routine is called when the push button is pressed. Each time the button is pressed, the lamp should switch from 'OFF' to 'ON', or from 'ON' to 'OFF'.

```
  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2019-09-30
; Author:
; Andrei Neagu
; Konstantinos Tatsis
;
; Lab number: 3
; Title: How to use interrupts
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Turn LED0 on and off with interrupts
;
; Input ports: PORTD
;
; Output ports: PORTB
;
; Subroutines: start, main, leds, delay
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00                               ;setup interrupt vectors
rjmp start                              ;

.org INT0addr
rjmp leds

.org 0x72                               ; program start
start:
ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address

ldi r16, 0x00                   ; set PORTD as input
out DDRD,r16

ldi r16, 0x01                   ; set PORTB1 as output
out DDRB, r16

ldi r16, 0b0000_0001    ;enable INT0, interrupt 0
out EIMSK, r16

ldi r16, 0b0000_0010    ;set EICRA(INT0-3) to to falling edge
sts EICRA, r16
sei                                             ;Sets the Global Interrupt flag (I) in SREG (status registe

ldi r16, 0b0000_0001    ;used for led display
main:
nop                                             ;wait
rjmp main

leds:
com r16
out PORTB, r16                  ; turn on LED0

ldi r22, 200                    ;my delay
delay:
```

```
dec r22
cpi r22,0
brne delay
reti
```

1.1 Flowchart 1



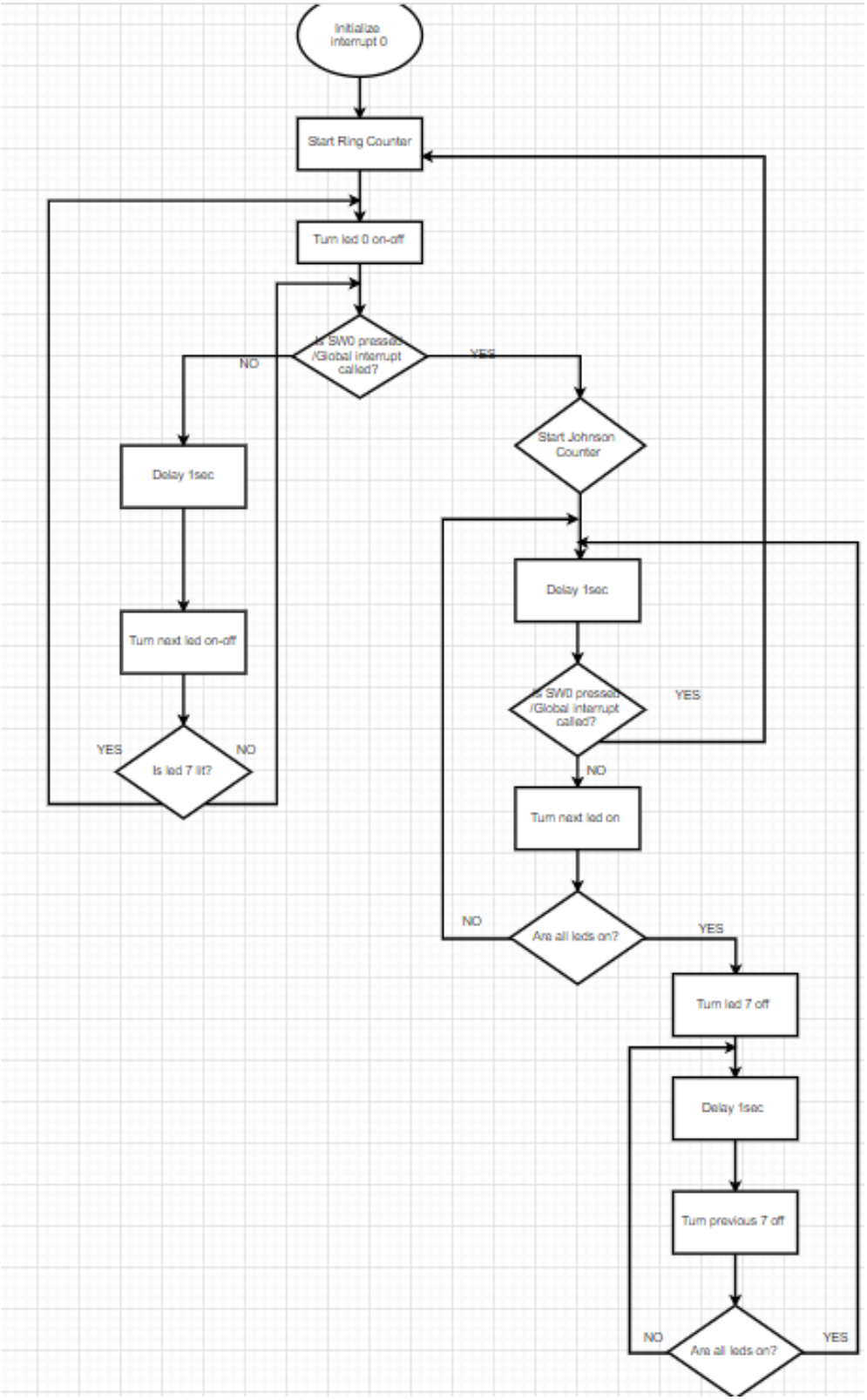
2 Task 2

Write a program that by means of a switch can choose to flash 8 LEDs either in the form of a ring counter or in the form of a Johnson counter. Use the switch SW0 connected to PORTD to switch between the two counters. Each time the button is pressed, a shift between the two counters should take place. By using interrupts you'll swap directly with no delay.

```
  ;>>>
```

```
                out PORTB, leds
                com leds
                lsl leds                                ;Shifts the bits to leds register
                com leds                                ;Complement/flip the value of leds
                rcall delay                             ;Call delay subroutine
rjmp ring                                               ;Go back to ring subroutine

        fixLedsOff:
                ldi leds, 0xFE
                rjmp ring

; JOHNSON COUNTER
johnson_on:
        cpi decider, 0x00       ;if the value is loaded then
        breq reset_ring         ; then branch to "reset_ring"

        cpi leds, 0x00
        breq johnson_off

        out PORTB, leds
        lsl leds
        rcall delay
rjmp johnson_on

johnson_off:
        cpi decider, 0x00
        breq reset_ring

        out PORTB, leds
        cpi leds, 0xFF
        breq johnson_on
        com leds                        ;Complement/flip the value of leds
        lsr leds                        ;"Logical shift to the right" shifts the value in register leds
        com leds
        rcall delay
rjmp johnson_off


; Generate by delay loop
delay:
        ldi r18, 3
        ldi r19, 138
        ldi r21, 86
L1: dec r21
        brne L1
        dec r19
        brne L1
        dec r18
        brne L1
        rjmp PC+1
ret     ;Return to where the call was made

interrupt_0:
        com decider     ;Flip the value of decider
        reti            ;Return from the interrupt

reset_ring:
        ldi leds, 0xFF
        out PORTB, leds
        rjmp ring

reset_john:
        ldi leds, 0xFF
        out PORTB, leds
        rjmp johnson_on
```

2.1 Flowchart 2



3 Task 3

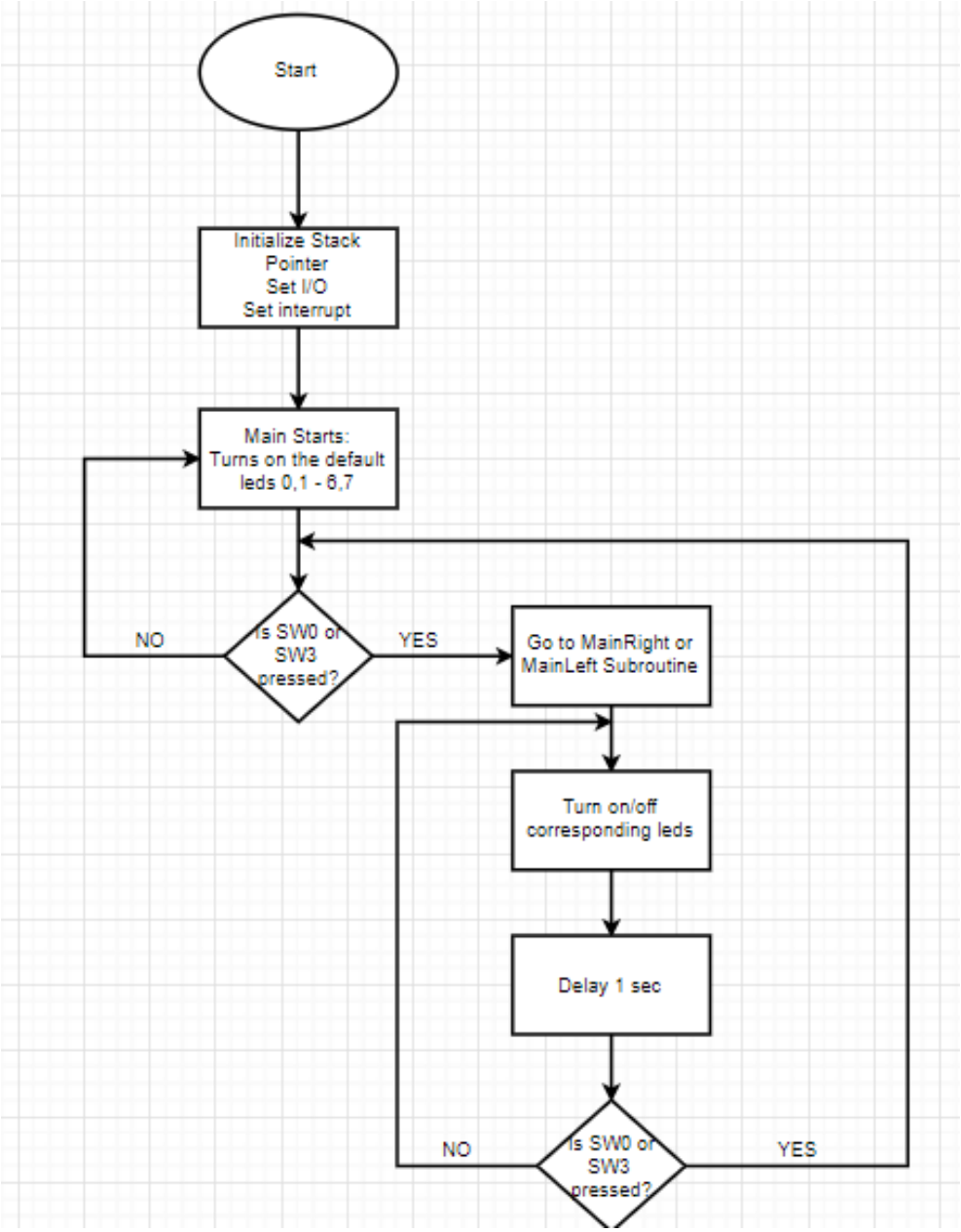
Interrupt.
Program that simulates the rear lights on a car The 8 LEDs should behave like the rear lights.
Function:
Normal light: LED 0, 1, 6 and 7 'ON'.
Turning right: LED 6 – 7 on, LED 0 – 3 blinking as RING counter.
Turning left: LED 0 – 1 on, LED 4 – 7 blinking as RING counter.

```
  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2019-10-02
; Author:
; Andrei Neagu
; Konstantinos Tatsis
;
; Lab number: 3
; Title: How to use interrupts
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Rear lights of a car
;
; Input ports: PORTD
;
; Output ports: PORTB
;
; Subroutines: start, main
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.def onleds = r22
.def leds = r21
.def LeftFlag = r24
.def RightFlag = r23

.org 0x00               ; constant to "store/load" to EIMSK address /set interrupt_0
rjmp start

.org INT0addr
rjmp interrupt_0
.org INT3addr
rjmp interrupt_3

.org 0x72               ; set interrupt_3

start:
ldi r16, 0b1111_1111    ;Load value "0b1111_1111" to register "r16"
out DDRB, r16                   ;Output value of "r16" register to PORTA
ldi RightFlag, 0b0000_0000      ;Load value 1 to RightFlag
ldi LeftFlag, 0b0000_0000
ldi R20, HIGH (RAMEND)
out SPH, R20

ldi R20, low (RAMEND)           ;Initialize Stack Pointer
out SPL, R20
ldi r16, 0b0000_1001
out EIMSK, r16
ldi r16, 0b1000_0010
sts EICRA, r16                          ;"Store Direct to data space" —— Load value of register to the "E
sei                                             ;Enables the "set Global interrupt flag"

Main:
ldi onleds, 0b0011_1100         ;Load value of 0b0011_1100 to onleds
out PORTB, onleds                       ;Light leds for "STATE 1"

checkMain:
cpi RightFlag, 0b1111_1111      ;If 0b1111_1111 is loaded to RightFlag
```

```
breq Right                              ;then go to "MainRight sub"
cpi LeftFlag, 0b1111_1111       ;If 0b1111_1111 is loaded to LeftFlag
breq Left                               ;then Go to "MainLeft"

rjmp checkMain                          ;Else jump to main / leave "STATE 1" lights on

interrupt_0:
com RightFlag                           ;Complement/flip the value of leds
clr LeftFlag                            ;Clear LeftFlag register
reti                                            ;Return from the "interrupt_0"

interrupt_3:
com LeftFlag                            ;Same as "interrupt_0"
clr RightFlag
reti


Left:
        ldi onleds, 0b0000_0011         ;Load value "0b0000_0011" to onleds

        loadCounterL:
        ldi r17, 0b0001_0000

        loopLeft:

        cpi LeftFlag, 0x00              ;If 1 is loaded to LeftFlag
        breq Main                               ;Branch to "Main subroutine"
        cpi RightFlag, 0xFF             ;Same for MainRight subroutine
        breq Right

        mov leds, onleds                ;Copy Register the value of "onleds" to leds
        add leds, r17                   ;Add the value of the register
        com leds                                ;Complement/flip the value of leds

        out PORTB, leds                 ;Light to PORTB the value of leds
        rcall delay                             ;Call delay

        lsl r17                                 ;Shift all bits to the left from value of r17
        cpi r17, 0b0000_0000    ;If value 1 is loaded to r17
        breq loadCounterL               ;Branch to "loadCounter"

        rjmp loopLeft


Right:
        ldi onleds, 0b1100_0000 ;SAME for "MainRight" as "MainLeft"

        loadCounter:
        ldi r17, 0b0000_1000

        loopright:
        cpi RightFlag, 0b0000_0000
        breq Main
        cpi LeftFlag, 0b1111_1111
        breq Left

        mov leds, onleds
        add leds, r17
        com leds

        out PORTB, leds
        rcall delay

        lsr r17
        cpi r17, 0b0000_0000
        breq loadCounter

        rjmp loopright

delay:

        ldi r18, 3
        ldi r19, 138
        ldi r20, 86
L1: dec r20
        brne L1
        dec r19
        brne L1
        dec r18
```

```
        brne  L1
        rjmp  PC+1
ret
```

3.1 Flowchart 3



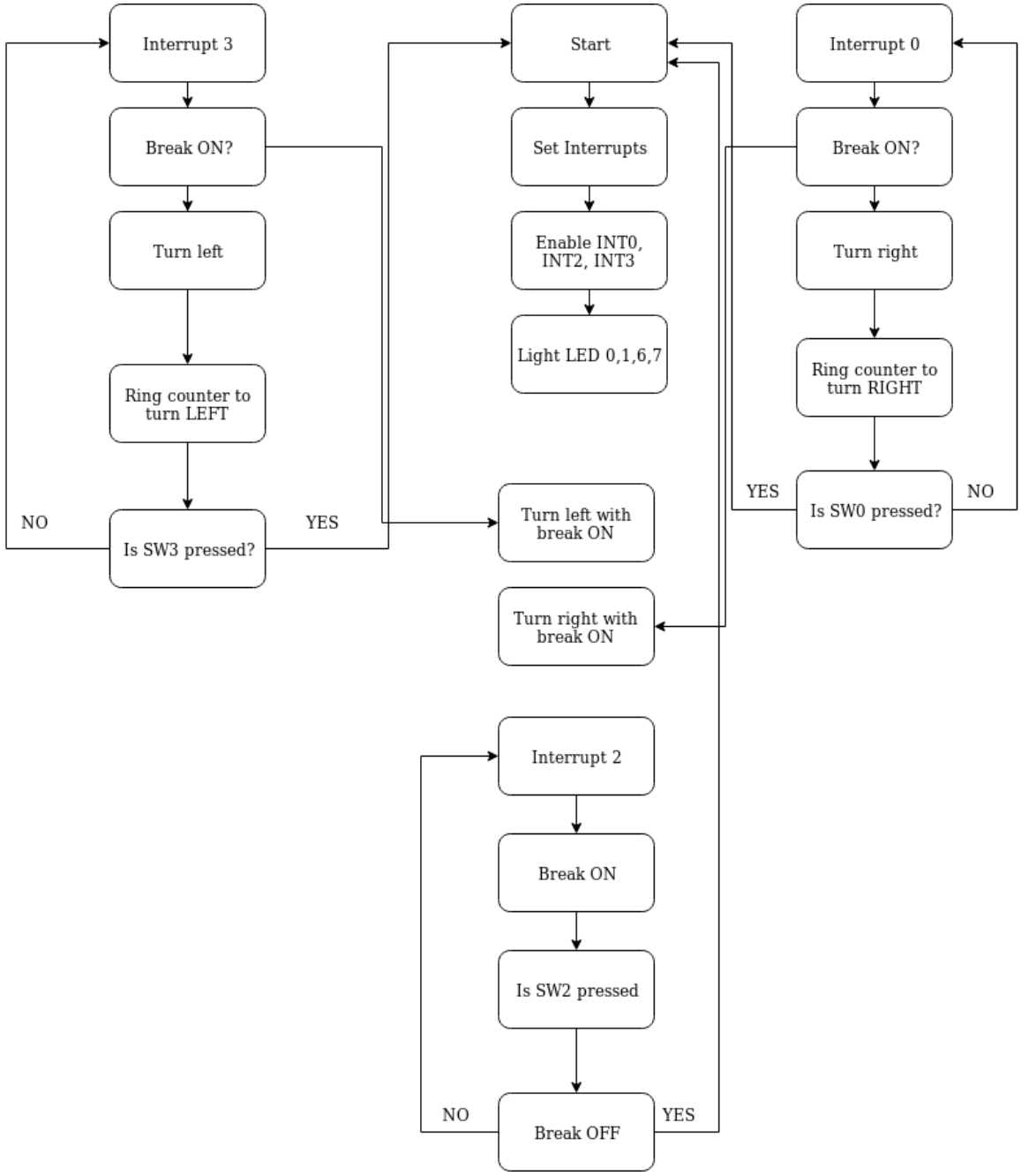
4 Task 4

Add function for the stop light to the previous task. When braking, all LEDs light up, if blink on the right or left is not going on. Turning right and brake: LED 4 – 7 on, LED 0 – 3 blinking as RING counter.
Turning left and brake: LED 0 – 3 on, LED 4 – 7 blinking as RING counter. Use INT2 for the Brake.

```
  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2019-10-02
; Author:
; Andrei Neagu
; Konstantinos Tatsis
;
; Lab number: 3
; Title: How to use interrupts
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Rear lights with a brake
;
; Input ports: PORTD
;
; Output ports: PORTB
;
; Subroutines: start, main, leds, delay
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.org 0x00                               ; setup interrupts
rjmp start

.org INT0addr
rjmp interrupt_0

.org INT2addr
rjmp interrupt_2

.org INT3addr
rjmp interrupt_3

.org 0x72
start:

ldi mr, 0b00001101              ; enable interrupt 0, 2, 3
out EIMSK, mr

ldi mr, 0b00000000              ; interrupt request setup
sts EICRA, mr

ldi mr , 0x00                   ; PORTD is set as an input
out DDRD, mr
sei                                             ; set global interrupt enable



ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address

.DEF mr = r16
.DEF mri = r17
.DEF flag1 = r22
.DEF flag2 = r23
.DEF flag3 = r24
.DEF flag4 = r25
```

```
on:
ser flag1                               ; Loads $FF directly to register
ser flag2                               ; Loads $FF directly to register
ser flag3                               ; Loads $FF directly to register
clr flag4                               ; Clear flag4

ldi mr,0xFF
out DDRB, mr
ldi r16, 0b00111100             ; LED0,1,6,7 are lit
out PORTB, mr
rjmp on

turnRight:
clr flag1
ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address

ldi mr , 0xFF
out DDRB, mr

RingCounter:

start1:
        ldi mri, 0b00110111 ; LED7,6 and 3 lit
        out PORTB, mri
        rcall delay
                ldi mr,  0b0000_1100
myloop:
        eor mri, mr                             ; exclusive or between mr and mri
        out PORTB,mri
        lsr mr                                  ; shift right the bits in mr

        cpi mri, 0b0011_1111    ; if equal do ring counter again
        breq RingCounter
        rcall delay
        rjmp myloop                             ; when this is reached do it again

turnLeft:
clr flag2
ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address
ldi mr , 0xFF
out DDRB, mr

RingCounter2:

start2:
        ldi mri, 0b1110_1100  ; LED4,1 and 0 lit
        out PORTB, mri
        rcall delay
        ldi mr,  0b0011_0000
myloop2:
        eor mri, mr                             ; exclusive or between mr and mri
        out PORTB,mri
        lsl mr                                  ; shift left the bits

        cpi mri, 0b1111_1100
        breq RingCounter2               ; if equal do ring counter again
        rcall delay
        rjmp myloop2                    ; when this is reached do it again



breakWhenOn:
clr flag3                                       ; used for leds
ser flag4                                       ;set to xFF
  ldi mr,0xFF
  out DDRB, mr
  out PORTB, flag3                      ; lights all leds
  rjmp breakWhenOn

turnLeftBreak:
clr flag4 //claring flag4
ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
```

```
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address
ldi mr , 0xFF
out DDRB, mr
RingWithBreak1:                 ; ring couter that starts at led 4 and goes left
start3:
        ldi mri, 0b1110_0000
        out PORTB, mri
        rcall delay
        ldi mr,  0b0011_0000
myloop3:
        eor mri, mr                     ; exclusive or between mri and  mr
        out PORTB,mri
        lsl mr                          ; shift bits to the left
        cpi mri, 0b1111_0000
        breq RingWithBreak1
        rcall delay
        rjmp myloop3


turnRightBreak:
clr flag4
ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address
ldi mr , 0xFF
out DDRB, mr

RingWithBreak2:
start4:
        ldi mri, 0b0000_0111 ; ring couter that starts at led 3 and goes right
        out PORTB, mri
        rcall delay
        ldi mr,  0b0000_1100
myloop4:
        eor mri, mr                     ; exclusive or between mri and  mr
        out PORTB,mri
        lsr mr
        cpi mri, 0b0000_1111
        breq RingWithBreak2
        rcall delay
        rjmp myloop4

interrupt_0:                    ; starts when button 0 is pressed goes right
sei
cpi flag4 , 0xff
breq turnRightBreak
cpi flag1 , 0xff
breq turnRightAux
brne idle

interrupt_2:                    ; starts when button 2 is pressed
sei
cpi flag3 ,0xff
breq breakWhenOn
brne idle

interrupt_3:                    ; starts when button 3 is pressed goes left
sei
cpi flag4 , 0xff
breq turnLeftBreak
cpi flag2 , 0xff
breq turnLeftAux
brne idle

turnRightAux:                   ; helper
        rjmp turnRight
turnLeftAux:                    ; helpe
        rjmp turnLeft

idle:                                   ; helper
        rjmp on

delay:
ldi  r18, 5
ldi  r19, 15
ldi  r20, 242
```

```
        L1: dec   r20
        brne L1
        dec   r19
        brne L1
        dec   r18
        brne L1
ret
```

4.1 Flowchart 4