Linnaeus University

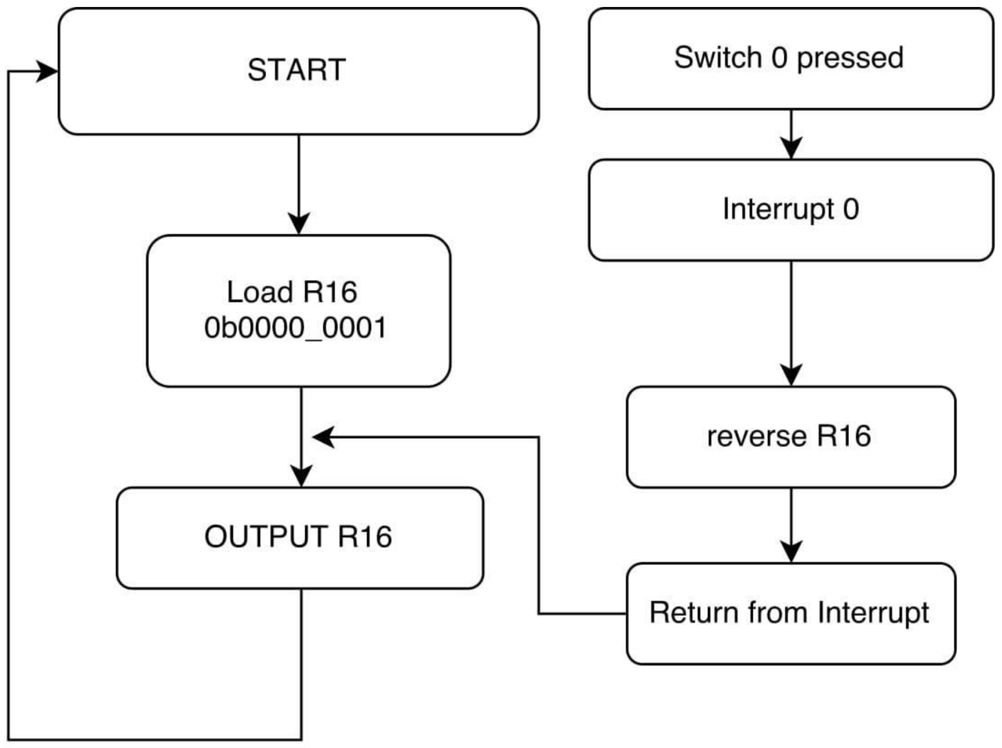
1DT301 - Computer Technology
Laboration 3

Students:
Alexander Risteski - ar222yu@student.lnu.se
Dimitrios Argyriou - da222kf@student.lnu.se

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'.

FLOWCHART:



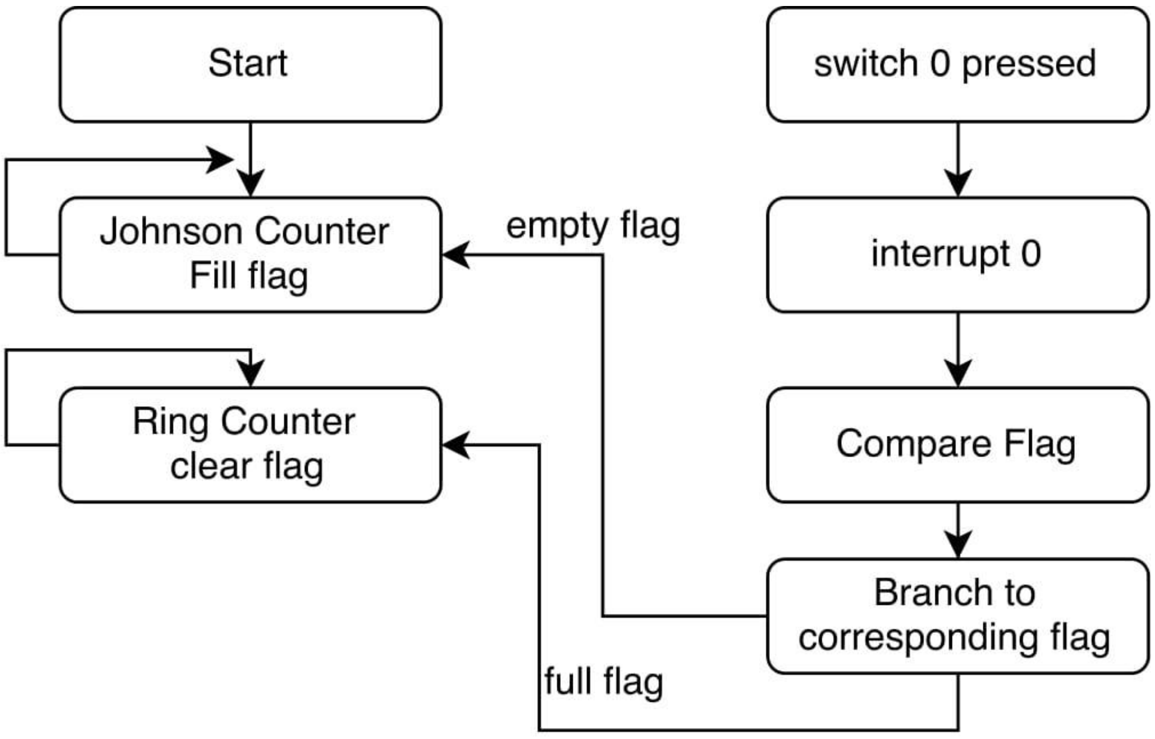
CODE:

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; 1DT301, Computer Technology I
; Date: 07/10 Date: 07/10 /2017
; Authors:
; Alexander Risteski
; Dimitrios Argyriou
;
; Hardware: STK600, CPU ATmega2560
; Function: This program lights on when off and off when on
;  led 0 when interrupt is called.
;
; Input ports: On-board switches connected to PORTD.
; Output ports: On -board LEDs connected to PORTB.
;
; Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

 .org 0x00
 rjmp start
 .org INT0addr
 rjmp interrupt_0

 .org 0x72
 start:
  ; Initialize SP, Stack Pointer
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
out DDRD,r16

ldi r16, 0x01
out DDRB, r16

ldi r16, 0b00000001
out EIMSK, r16

ldi r16, 0b00000100
sts EICRA, r16
sei

;main program
ldi r16, 1
main_program:
nop
rjmp main_program

interrupt_0:
com  r16
out PORTB, r16

ldi r22, 200
delay_int:
dec r22
cpi r22,0
brne delay_int

reti
```

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.

FLOWCHART:



CODE:

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; 1DT301, Computer Technology I
; Date: 07/10 Date: 07/10 /2017
; Authors:
; Alexander Risteski
; Dimitrios Argyriou
;
; Hardware: STK600, CPU ATmega2560
; Function: This program switches from ring counter to Johnson counter
; by calling interrupt (INT0addr) on switch 0
;
; Input ports: On-board switches connected to PORTD.
; Output ports: On -board LEDs connected to PORTB.
;
; Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
```

```
.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0



.org 0x72
start:

ldi mr, 0b00000011
out EIMSK, mr

ldi mr, 0b00001000
sts EICRA, mr

ldi mr , 0x00
out DDRD, mr

sei

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

; Assigning names to the registers
.DEF mr = r16
.DEF mri = r17
.DEF flag =r23
;=====================Johnson counter============================

JohnsonCounter:
ser flag
ldi mr, 0xff
out DDRB, mr

forward:

out PORTB, mr
lsl mr
call delay
cpi mr, 0b00000000
brne forward
rjmp reset

reset:
out PORTB,mr
```

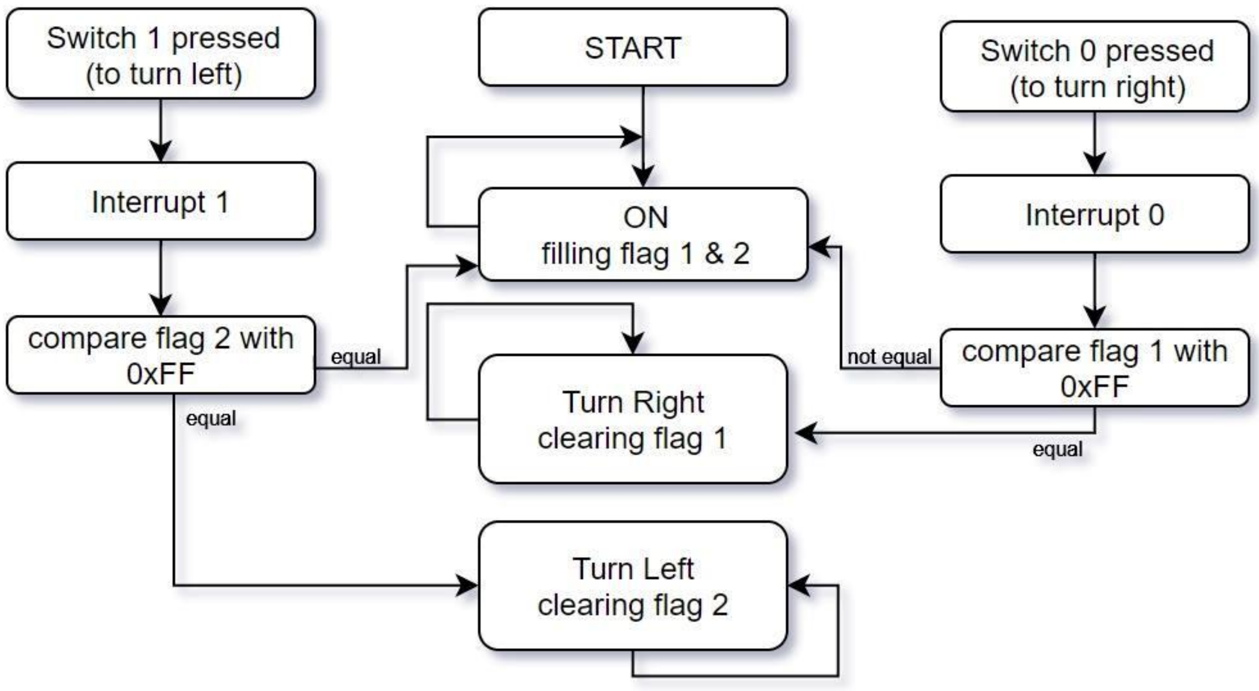
```
call delay
ldi mri,0b10000000
rjmp backwards

backwards:

lsr mr
add mr, mri
out PORTB,mr
call delay
cpi mr, 0xFF
brne backwards

rjmp JohnsonCounter

;======================= RING COUNTER ================================
RingCounter:
clr flag
start1:
ldi mr, 0x01
com mr                          ; complements the rergister so that it will be showed correctly
out PORTB, mr
com mr                          ;complements again to return to its original form
rcall delay

myloop:


lsl mr
com mr
out PORTB, mr
com mr

cpi mr, 0b00000000
breq equal
rcall delay
rjmp myloop

equal:
rjmp RingCounter

interrupt_0:

ldi r22,200
delay_int:
            dec r22
            cpi r22,0
            brne delay_int
            sei
```

```
cpi flag, 0xff
breq RingCounter
brne JohnsonCounter

delay:
ldi  r18, 5
ldi  r19, 15
ldi  r20, 242
L1: dec  r20
brne L1

dec  r19
brne L1

dec  r18
brne L1

ret
```

TASK 3

Rear lights on a car 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.

FLOWCHART:



CODE:

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; 1DT301, Computer Technology I
; Date: 07/10 Date: 07/10 /2017
; Authors:
; Alexander Risteski
; Dimitrios Argyriou
;
; Hardware: STK600, CPU ATmega2560
; Function: This program  simulates the rear lights on a car.
; It has 3 states and changes between them by calling interrupt_0 and _1
; ON (Leds 7,8 & 0,1).
;Turn right :Leds 6 – 7 on, led 0 –3 blinking as RING counter.
;Turn left :led 0 – 1 on, led 4 – 7 blinking as RING counter.
;
; Input ports: On-board switches connected to PORTD.
; Output ports: On -board LEDs connected to PORTB.
;
; Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
```

```
.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0

.org INT1addr
rjmp interrupt_1

.org 0x72

start:

ldi mr, 0b00000011
out EIMSK, mr

ldi mr, 0b00001000
sts EICRA, mr

ldi mr , 0x00
out DDRD, mr

sei

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
;========================== ON ===============================
; Filling up the flags1,2 so that it can branch to one of the states
; and when branches clear the appropriate flag so that it can branch here again.
on:
ser flag1
ser flag2
  ldi mr,0xFF
  out DDRB, mr
  ldi r16, 0b00111100
  out PORTB, mr
  rjmp on

;================== RIGHT TURN ===============================
  ; Initialize SP, Stack Pointer
turnRight:
clr flag1 // clear the flag1
ldi r20, HIGH(RAMEND)      ; R20 = high part of RAMEND address
```

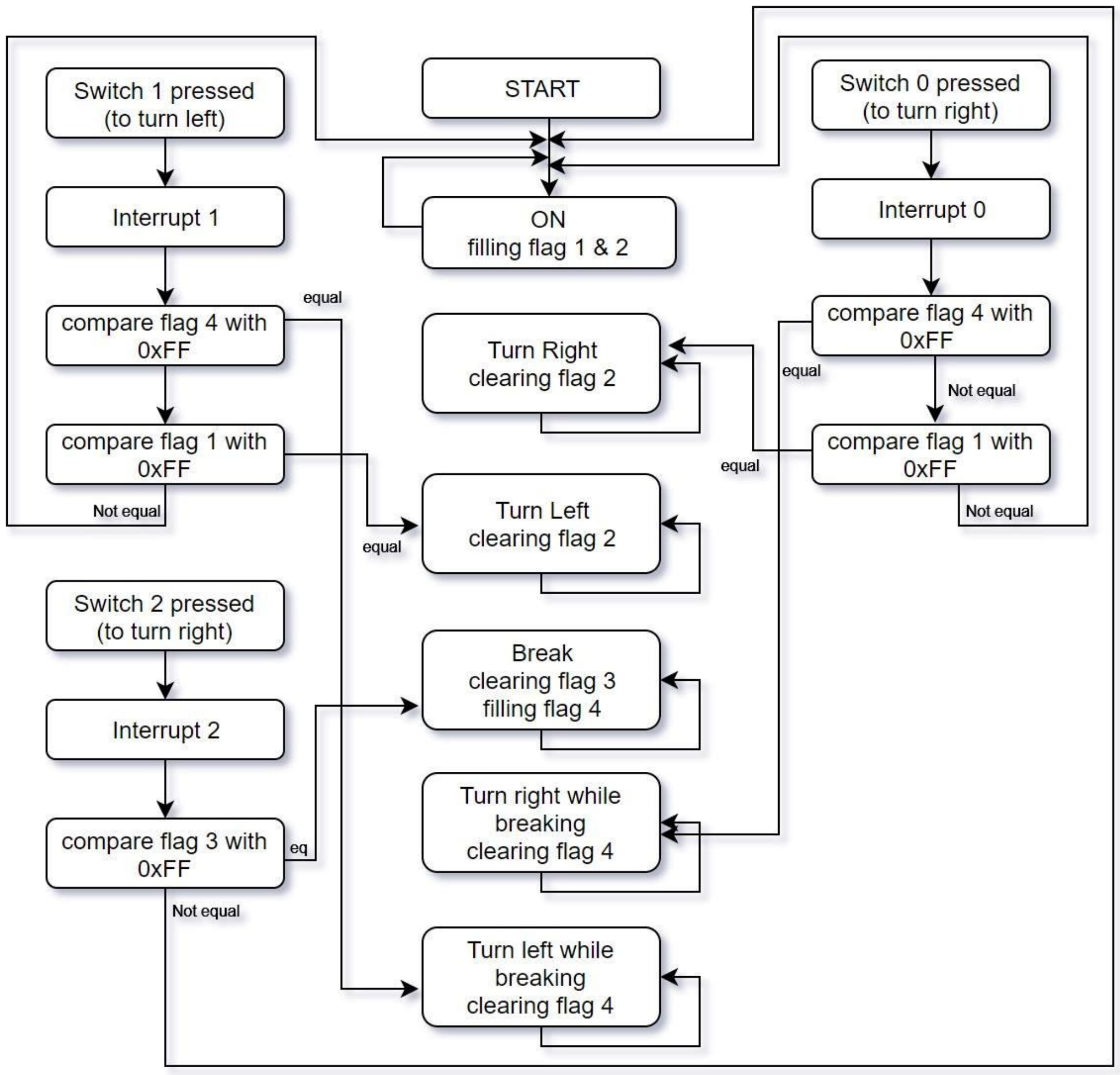
```
out SPH,R20                ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)       ; R20 = low part of RAMEND address
out SPL,R20                ; SPL = low part of RAMEND address

ldi mr , 0xFF
out DDRB, mr

RingCounter:

start1:
            ldi mri, 0b00110111



            out PORTB, mri
            rcall delay
            ldi mr,         0b0000_1100
myloop:
            eor mri, mr
            out PORTB,mri
            lsr mr

            cpi mri, 0b0011_1111
            breq RingCounter
            rcall delay
            rjmp myloop

;================== LEFT TURN ===============================
             ; Initialize SP, Stack Pointer
turnLeft:
clr flag2 // clear the 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
            out PORTB, mri
            rcall delay
            ldi mr,         0b0011_0000
myloop2:

            eor mri, mr
            out PORTB,mri
            lsl mr
```

```
                cpi mri, 0b1111_1100
                breq RingCounter2
                rcall delay
                rjmp myloop2

====================== INTERRUPT SUBROUTINES ============================
interrupt_0:
sei // set interrupt enabled
cpi flag1, 0xff
breq turnRight
brne on

interrupt_1:
sei // set interrupt enabled
cpi flag2, 0xff
breq turnLeft
brne 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
```

TASK 4

Rear lights on a car, with light for brakes Add function for the stop light to the previous task. When braking, all LED slight 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.

FLOWCHART:


START
ON filling flag 1 & 2
Switch 1 pressed (to turn left)
Interrupt 1
compare flag 4 with 0xFF
compare flag 1 with 0xFF
Not equal
equal
equal
Switch 0 pressed (to turn right)
Interrupt 0
compare flag 4 with 0xFF
equal
Not equal
compare flag 1 with 0xFF
equal
Not equal
Turn Right clearing flag 2
Turn Left clearing flag 2
Switch 2 pressed (to turn right)
Interrupt 2
compare flag 3 with 0xFF
eq
Not equal
Break clearing flag 3 filling flag 4
Turn right while breaking clearing flag 4
Turn left while breaking clearing flag 4

CODE:

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; 1DT301, Computer Technology I
; Date: 07/10 Date: 07/10 /2017
; Authors:
; Alexander Risteski
; Dimitrios Argyriou
;
```

```
; Hardware: STK600, CPU ATmega2560
; Function: This program  simulates the rear lights on a car from task3 plus
; introducing more states for breaking.
; Breaking : all LEDS on
; Breaking and turning right :LED 4 – 7 on, LED 0 – 3 blinking as RING counter..
; Breaking and turning left :LED 0 – 1 on, led 4 – 7 blinking as RING counter.
;
; Input ports: On-board switches connected to PORTD.
; Output ports: On -board LEDs connected to PORTB.
;
; Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0

.org INT1addr
rjmp interrupt_1

.org INT2addr
rjmp interrupt_2

.org 0x72

start:

ldi mr, 0b00000111
out EIMSK, mr

ldi mr, 0b00001000
sts EICRA, mr

ldi mr , 0x00
out DDRD, mr

sei

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

;====================== DEFINING REGISTERS ==============================
.DEF mr = r16
.DEF mri = r17
```

```
.DEF flag1 = r22
.DEF flag2 = r23
.DEF flag3 = r24
.DEF flag4 = r25
;==================================== ON =======================================
;============= Filling up flags1,2,3 with 0xff ================================
on:
ser flag1
ser flag2
ser flag3
clr flag4 // clear flag4

;============== ON state (Leds 7,8,0,1 are on)=================================
  ldi mr,0xFF
  out DDRB, mr
  ldi r16, 0b00111100
  out PORTB, mr
  rjmp on

;============================== RIGHT TURN ====================================
  ; Initialize SP, Stack Pointer
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
            out PORTB, mri
            rcall delay
                        ldi mr,       0b0000_1100
myloop:
            eor mri, mr
            out PORTB,mri
            lsr mr

            cpi mri, 0b0011_1111
            breq RingCounter
            rcall delay
            rjmp myloop

;============================= LEFT TURN =====================================
             ; Initialize SP, Stack Pointer
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
                out PORTB, mri
                rcall delay
                ldi mr,         0b0011_0000
myloop2:
                eor mri, mr
                out PORTB,mri
                lsl mr

                cpi mri, 0b1111_1100
                breq RingCounter2
                rcall delay
                rjmp myloop2
;======================= Helper subroutines for branching====================
                turnRightBridge:
                rjmp turnRight
                turnLeftBridge:
                rjmp turnLeft
                turnOnBridge:
                rjmp on
;========================= BREAK ==========================================
;======== When breaks is on, all lights are on ================================
breakWhenOn:
clr flag3 // Clearing flag3
ser flag4 // Filling flag4 to xFF
  ldi mr,0xFF
  out DDRB, mr
  ldi r16, 0x00
  out PORTB, mr
  rjmp breakWhenOn

;======================== BREAK LEFT=========================================
turnLeftBreak:
clr flag4 //claring flag4
 ; Initialize SP, Stack Pointer
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:
start3:
                ldi mri, 0b1110_0000


                out PORTB, mri
                rcall delay
                ldi mr,         0b0011_0000
myloop3:
                eor mri, mr
                out PORTB,mri
                lsl mr
                cpi mri, 0b1111_0000
                breq RingWithBreak1
                rcall delay
                rjmp myloop3


;========================= BREAK RIGHT =====================================
turnRightBreak:
clr flag4 //Clearing flag4
 ; Initialize SP, Stack Pointer
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
.DEF mr = r16
                .DEF mri = r17
ldi mr , 0xFF
out DDRB, mr
RingWithBreak2:
start4:
                ldi mri, 0b0000_0111 ;OBS!!! ob1110_0000 for task 4 break 00000111
                out PORTB, mri
                rcall delay
                ldi mr,         0b0000_1100
myloop4:
                eor mri, mr
                out PORTB,mri
                lsr mr

                cpi mri, 0b0000_1111
                breq RingWithBreak2
                rcall delay
                rjmp myloop4

;========================= INTERRUPT SUBROUTINES ============================
interrupt_0:
```

```
sei
cpi flag4, 0xff
breq turnRightBreak
cpi flag1, 0xff
breq turnRightBridge
brne turnOnBridge

interrupt_1:
sei
cpi flag4, 0xff//0x00
breq turnLeftBreak
cpi flag2, 0xff
breq turnLeftBridge
brne turnOnBridge

interrupt_2:
sei
cpi flag3 ,0xff
breq breakWhenOn
brne turnOnBridge
;============================== DELAY SUBROUTINE ============================
delay:
ldi  r18, 5
ldi  r19, 15
ldi  r20, 242
L1: dec  r20
brne L1
dec  r19
brne L1
dec  r18
brne L1
ret
```