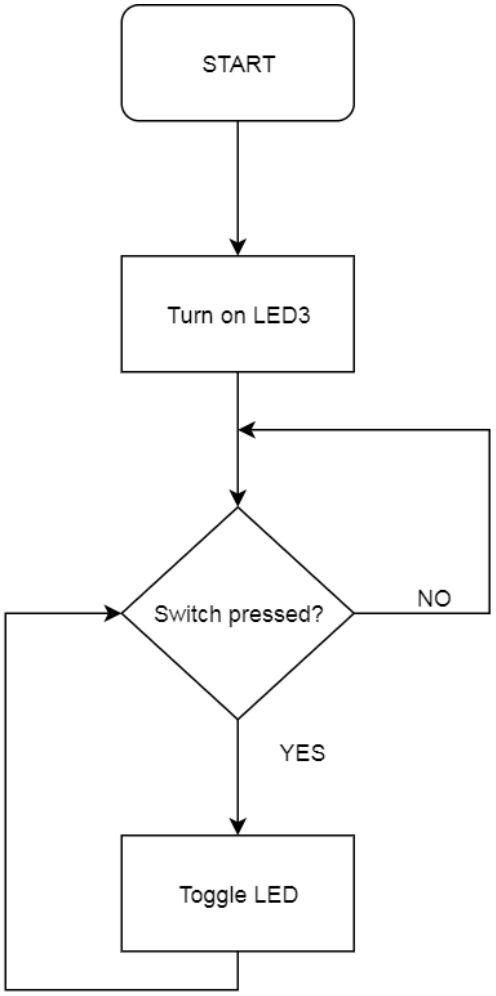
Task 1

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-09-25
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 3
; Title: Interrupts.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Turning ON and OFF a LED with a push button.
;
; Input ports: On-board switches connected to DDRD.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: interrupt_0, interrupt_1
; Included files: m2560def.inc
;
; Other information: Clock set at 1 MHz
;
; Changes in program: None.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include"m2560def.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0
.org INT1addr
rjmp interrupt_1

.org 0x72

start:
ldi r16, HIGH(RAMEND)          ; MSB part av address to RAMEND
out SPH, r16                   ; store in SPH
ldi r16, LOW(RAMEND)           ; LSB part av address to RAMEND
out SPL, r16                   ; store in SPL

ldi r20, 0x00
out DDRD, r20                  ; all zero's to DDRD, input
```

```
ldi r20, 0xFF
out DDRB, r20                ; all one's to DDRB, outputs

ldi r20, 0x03                ; INT0 and INT1 enabled
out EIMSK, r20

ldi r20, 0x08                ; INT1 falling edge, INT0 rising edge
sts EICRA, r20

sei                          ; global interrupt enable

ldi r20, 0xF7
out PORTB, r20

main:
nop
rjmp main

interrupt_0:
interrupt_1:
andi r20, 0x08               ; isolating the bit corresponding to LED3
com r20                      ; inverting the bit
out PORTB, r20               ; outing to LED3
reti                         ; return from interrupt
```



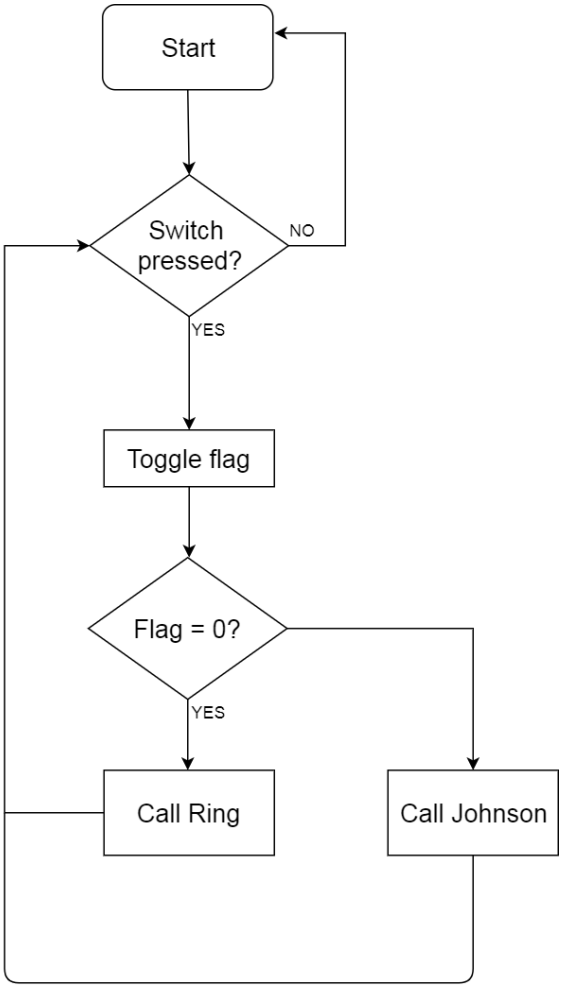

```
/* Description:
* The program will start with lighting up a LED that will alternatively turn Off/On when
* when using the interrupt (pressing the designated switch)
*/
```

Task 2

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-09-25
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 3
; Title: Interrupts.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: A program that switches between Ring counter and Johnson counter
; by using Interrupts
```

```
; Input ports: On-board switches connected to DDRD.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: interrupt_0, interrupt_1
; Included files: m2560def.inc
;
; Other information: Clock set at 1 MHz.
;
; Changes in program: 2017-09-26.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp interrupt_0
.org INT1addr
rjmp interrupt_1

.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 r20, 0xFF
out DDRB, r20                   ; All one's to DDRB, outputs

ldi r16, 0xFE                   ; starting with LED0
ldi r17, 0x00                   ; temp register to help with sloop

ldi r23, 0x00
out DDRD, r23

ldi r22, 0x00                   ; keep track of counters, if 0 then ring, else johnson

;could not test with SW0 because there were no available microcontrollers that had a functioning
SW0
ldi r24, 0x03                   ; INT0 and INT1 enabled
out EIMSK, r24

ldi r24, 0x08                ; INT1 falling edge, INT0 rising edge
```

```
sts EICRA, r24

sei                             ; global interrupt enable

main:
        cpi r22, 0x00
        breq ring
        rjmp johnsonLeft
rjmp main

ring:
        cpi r16, 0xFF           ; checking if all LEDs are off
        breq equal
        out PORTB, r16          ; write in PORTB, turning on LEDs
        com r16                 ; inverting the bits of r16
        lsl r16                 ; pushing a 0 to the left
        com r16                 ; inverting the bits of r16 again
        rcall delay
        rjmp ring

johnsonLeft:
cpi r22, 0x00
breq ring
cpi r16, 0x00                   ; check if all LEDs are on
breq johnsonRight
out PORTB, r16                  ; write to PORTB
lsl r16                         ; pushing 0 to the left to turn on next light aswell
rcall delay
rjmp johnsonLeft

johnsonRight:
out PORTB, r16                      ; write to PORTB
cpi r16, 0xFF                       ; check if all LEDs are off
breq johnsonLeft
mov r17, r16                        ; move r16's bits to r17
com r17                             ; invert r17's bits
lsr r17                             ; pushing 0 to the right
com r17                             ; invert r17's bits again
mov r16, r17                        ; move r17's bits to r16
rcall delay
rjmp johnsonRight

equal:
ldi r16, 0xFE

rjmp main

; Generated by delay loop calculator
```



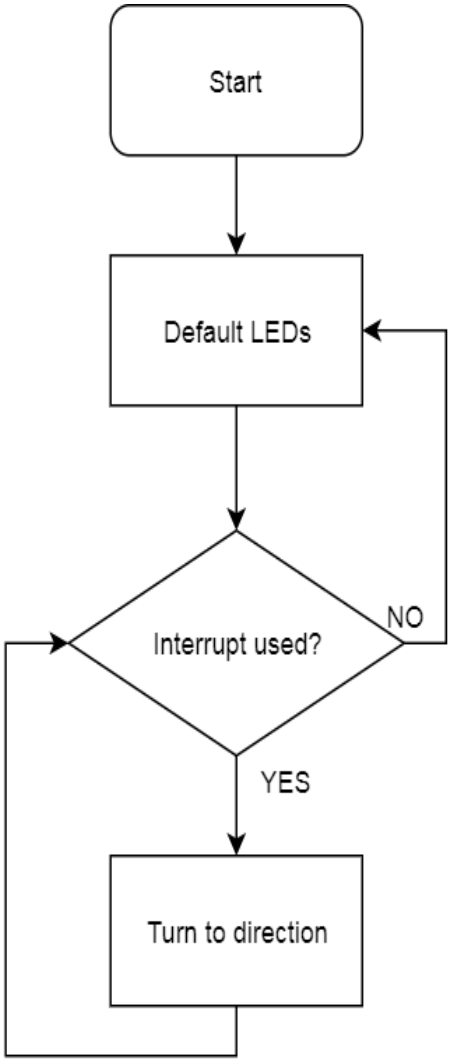
```
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 500 000 cycles
; 500ms at 1 MHz

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

interrupt_0:
interrupt_1:
        ldi r16, 0xFF
        out PORTB, r16          ; turn off all LEDs, to stop the counter
        com r22                 ; invert r22 to change counter
reti

/*Description
* The program switches between a Ring and a Johnson counter each time
*the interrupt has been used.
*/
```

Task 3

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-09-25
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 3
; Title: Interrupts.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: A program that simulates the blinking lights of a vehicle
; when trying to turn right/left
```

```
;
; Input ports: On-board switches connected to DDRD.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: interrupt_1, interrupt_2
; Included files: m2560def.inc
;
; Other information:  Clock set at 1MHz.
;
; Bouncing issues regarding the pressing of the
; switches when trying to activate the turn signals

;
; Changes in program: 2017-09-26, 2017-09-28, 2017-09-29
; 2017-10-01 Different updates and changes to code to get it
; to work properly
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include"m2560def.inc"

.org 0x00
rjmp start

.org INT1addr
jmp interrupt_1
.org INT2addr
jmp interrupt_2

.org 0x72

start:
ldi r16, HIGH(RAMEND)          ; MSB part av address to RAMEND
out SPH, r16                   ; store in SPH
ldi r16, LOW(RAMEND)           ; LSB part av address to RAMEND
out SPL, r16                   ; store in SPL

ldi r20, 0xFF
out DDRB, r20                  ; all one's to DDRB, output

ldi r20, 0x00
out DDRD, r20                  ; all zero's to DDRD, input

ldi r30, 0x00                  ; help register for interrupt_1
ldi r31, 0x00                  ; help register for interrupt_2

ldi r20, 0x06                  ; INT0 and INT1 enabled
out EIMSK, r20
```

```
ldi r20, 0x28                   ; INT1 falling edge, INT0 rising edge
sts EICRA, r20

sei                             ; global interrupt enable

main:
ldi r24, 0x3C
out PORTB, r24                  ; default
rjmp main

start_turning_right:
clr r16                         ; clear Stack
sei                             ; enable Interrupt
cpi r30, 0xFF                   ; check if interrupt was used again
brne main
ldi r22, 0x37                   ; 00xx 0xxx
        out PORTB, r22
        ldi r23, 0xA0           ; help register
        tright:
           out PORTB, r22       ; outputting first to not skip initial position
           cpi r22, 0x3F        ; checking if the ring is complete
           breq equal2
           com r22              ; inverting the bits
           lsr r22              ; pushing one's from the left side
           com r22              ; inverting again
           eor r22,r23          ; XOR to get higher nibble back to 0011
        rcall delay
        rjmp tright             ; loop it

        equal2:
                ldi r22, 0x37   ; reset to initial position
        rjmp tright

start_turning_left:
clr r16                         ; clear Stack
sei                             ; enable interrupts
cpi r31, 0xFF                   ; check if interrupt was used
brne main
ldi r25, 0xEC                   ; xxx0 xx00
        out PORTB, r25

        ldi r17, 0x05           ; help register

        tleft:
           out PORTB, r25       ; outputting first to not skip initial position
           cpi r25, 0xFC        ; checking if the ring is complete
           breq equal
```



```
            com r25                 ; inverting the bits
            lsl r25                 ; pushing one's from the right side
            com r25                 ; inverting again
            eor r25,r17             ; XOR to get lower nibble back to 1100
        rcall delay
        rjmp tleft                  ; loop it

        equal:
                ldi r25, 0xEC       ; reset to initial position
        rjmp tleft


interrupt_1:
com r30                         ; inverting r30
rjmp start_turning_right

reti                            ; return from interrupt

interrupt_2:
com r31                         ; inverting r31
rjmp start_turning_left

reti                            ; return from interrupt

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 500 000 cycles
; 500ms at 1 MHz

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

/*Description
*The program simulates the turning signals a car uses. It starts with the lights in default position
*0x3C, that will change depending which interrupt is being used. If interrupt1 is used then the left
side of the upper nibble will remain in position 00xx, while the lower nibble will be doing a ring
```

*counter to the right. When interrupt2 is used then the right side of the lower nibble will remain in
*position xx00, while the upper nibble will be doing a ring counter going to the left.
*Reusing the same interrupt while in the respective sequence will return the LEDs to the default
*position of 00xx xx00.
*/

Task 4

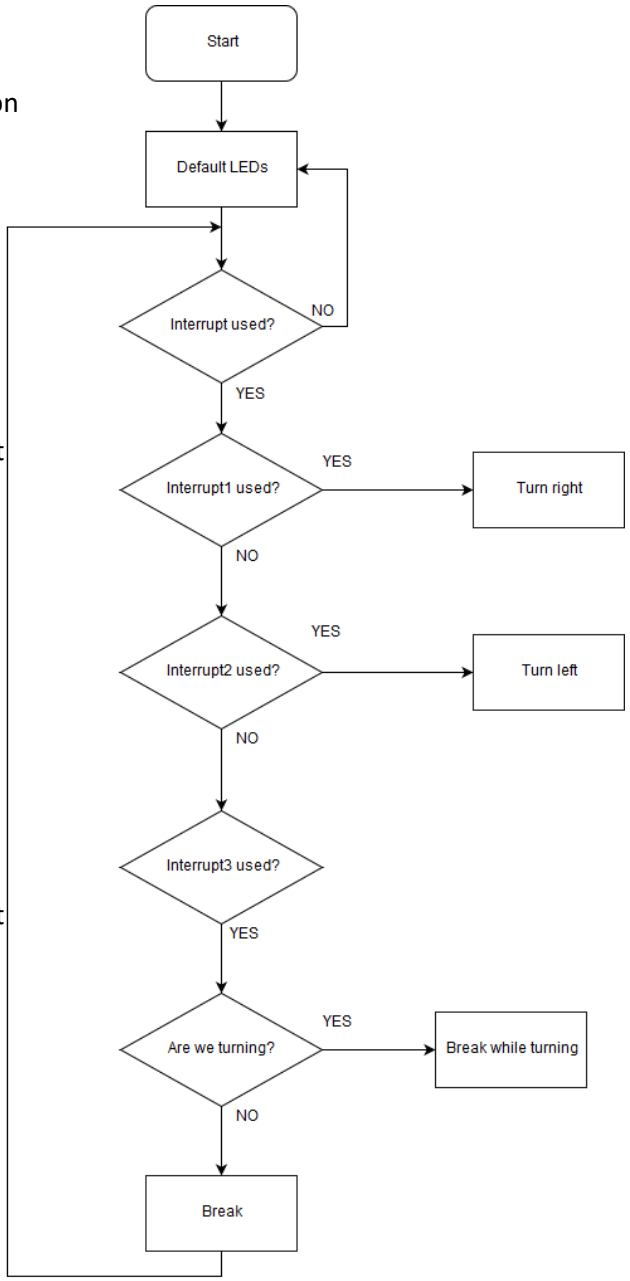
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-09-25
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 3
; Title: Interrupts.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: A program that simulates the blinking lights of a vehicle
; when trying to turn right/left
;
; Input ports: On-board switches connected to DDRD.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: interrupt_1, interrupt_2
; Included files: m2560def.inc
;
; Other information:  Clock set at 1MHz.
;
; Bouncing issues regarding the pressing of the
; switches when trying to activate the turn signals

;
; Changes in program: 2017-09-26, 2017-09-28, 2017-09-29
; 2017-10-01,2017-10-10 Different updates and changes to code to get it
; to work properly
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include"m2560def.inc"
.org 0x00
rjmp start

.org INT1addr
jmp interrupt_1
```

```
.org INT2addr
jmp interrupt_2
.org INT3addr
jmp interrupt_3

.org 0x72

start:
ldi r16, HIGH(RAMEND)          ; MSB part av address to RAMEND
out SPH, r16                   ; store in SPH
ldi r16, LOW(RAMEND)           ; LSB part av address to RAMEND
out SPL, r16                   ; store in SPL

ldi r20, 0xFF
out DDRB, r20                  ; all one's to DDRB, output

ldi r20, 0x00
out DDRD, r20                  ; all zero's to DDRD, input

ldi r30, 0x00                  ; help register for interrupt_1
ldi r31, 0x00                  ; help register for interrupt_2
ldi r29, 0x00                  ; help register for interrupt_3

ldi r23, 0xA0                  ; help with eor right
ldi r17, 0x05                  ; help with eor left
ldi r26, 0x11                  ; help with eor left 2
ldi r28, 0x88                  ; help with eor right 2


ldi r20, 0x0E                  ; INT0 and INT1 enabled
out EIMSK, r20

ldi r20, 0xE8                  ; INT2, INT1 falling edge, INT3 both rising and falling
sts EICRA, r20

sei                            ; global interrupt enable

main:
ldi r24, 0x3C
out PORTB, r24                 ; default
rjmp main

start_turning_right:
clr r16                        ; clear Stack
sei                            ; enable Interrupt
cpi r30, 0xFF                  ; check if interrupt was used again
brne main
ldi r22, 0x37                  ; 00xx 0xxx
```

```
        out PORTB, r22
        tright:
                out PORTB, r22          ; outputting first to not skip initial position
                cpi r22, 0x3F           ; checking if the ring is complete
                breq equal2
                com r22                 ; inverting the bits
                lsr r22                 ; pushing one's from the left side
                com r22                 ; inverting again
                eor r22,r23             ; XOR to get higher nibble back to 0011
        rcall delay
        rjmp tright                     ; loop it

        equal2:
                ldi r22, 0x37           ; reset to initial position
        rjmp tright

start_turning_left:
clr r16                                 ; clear Stack
sei                                     ; enable interrupts
cpi r31, 0xFF                           ; check if interrupt was used
brne main
ldi r25, 0xEC                           ; xxx0 xx00
        out PORTB, r25
        tleft:
                out PORTB, r25          ; outputting first to not skip initial position
                cpi r25, 0xFC           ; checking if the ring is complete
                breq equal
                com r25                  ; inverting the bits
                lsl r25                  ; pushing one's from the right side
                com r25                  ; inverting again
                eor r25,r17              ; XOR to get lower nibble back to 1100
        rcall delay
        rjmp tleft                       ; loop it

        equal:
                ldi r25, 0xEC           ; reset to initial position
        rjmp tleft


breaks:
clr r16                                 ; clear Stack
sei                                     ; enable interrupts
cpi r29, 0xFF
breq checkpos
                                        ; not sure

checkpos:
cpi r30, 0xFF                           ; check if it's turning right
```

```
breq break_right
cpi r31, 0xFF                   ; check if it's turning left
breq break_left
rjmp break_def                  ; check if it's in default state

; break in default state
break_def:
clr r16                         ; clear Stack
sei                             ; enable interrupts
ldi r24, 0x00
out PORTB, r24                  ; turn on all LEDs
rjmp checkpos

; break when turning right
break_right:
clr r16                         ; clear Stack
sei                             ; enable Interrupt
/*cpi r29, 0xFF                 ; check if interrupt was used again
breq start_brk_right*/
start_brk_right:
ldi r22, 0x07                   ; 0000 0xxx
        out PORTB, r22
        breakright:
                out PORTB, r22
                cpi r22, 0x0F           ; checking if the ring is complete
                breq equal2brk
                com r22                 ; inverting the bits
                lsr r22                 ; pushing one's from the left side
                com r22                 ; inverting again
                eor r22,r28             ; XOR to get higher nibble back to 0000;

        rcall delay
        rjmp breakright                  ; loop it

equal2brk:
        ldi r22, 0x07                   ; reset to initial position
        rjmp breakright


break_left:
clr r16                         ; clear Stack
sei                             ; enable interrupts
/*cpi r29, 0xFF                 ; check if interrupt was used
breq start_brk_left
rjmp main*/

start_brk_left:
ldi r25, 0xE0                    ; xxx0 0000
```

```
        out PORTB, r25
        breakleft:
                out PORTB, r25
                cpi r25, 0xF0           ; checking if the ring is complete
                breq equalbrk
                com r25                 ; inverting the bits
                lsl r25                 ; pushing one's from the right side
                com r25                 ; inverting again
                eor r25, r26            ; XOR to get lower nibble to 0000

        rcall delay
        rjmp breakleft                  ; loop it

equalbrk:
        ldi r25, 0xE0                   ; reset to initial position
        rjmp breakleft



interrupt_1:
com r30                                 ; inverting r30
rjmp start_turning_right

reti                                    ; return from interrupt

interrupt_2:
com r31                                 ; inverting r31
rjmp start_turning_left

reti

interrupt_3:
com r29
rjmp breaks

reti                                    ; return from interrupt

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 500 000 cycles
; 500ms at 1 MHz

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

/*Description
*The program simulates the lights on a car, when turning to left/right, breaking and breaking while
*turning left/right.
*The default LEDs are in position 00xx xx00. When interrupt1 or interrupt2 is used the LEDs will start
*to blink in a ring counter going right, respectively left. However, if interrupt3 is used then it will
turn *on all LEDs on the upper, respective lower nibble (upper if turning right, lower if turning left). If
the *LEDs are in default position and interrupt3 is used then all 8 LEDs will light up.
*/
```