Computer Technology

Report for Lab 3

Student: Helena TEVAR HERNANDEZ
Student ID: ht222fd@student.lnu.se
Student: Jiahui LE
Student ID: jl224bn@student.lnu.se
Semester: HT2018
Area: Computer Technology
Course code: 1DV607

Contents

1 Task 1

Switch A LED With the Interrupt

1.1 Diagram

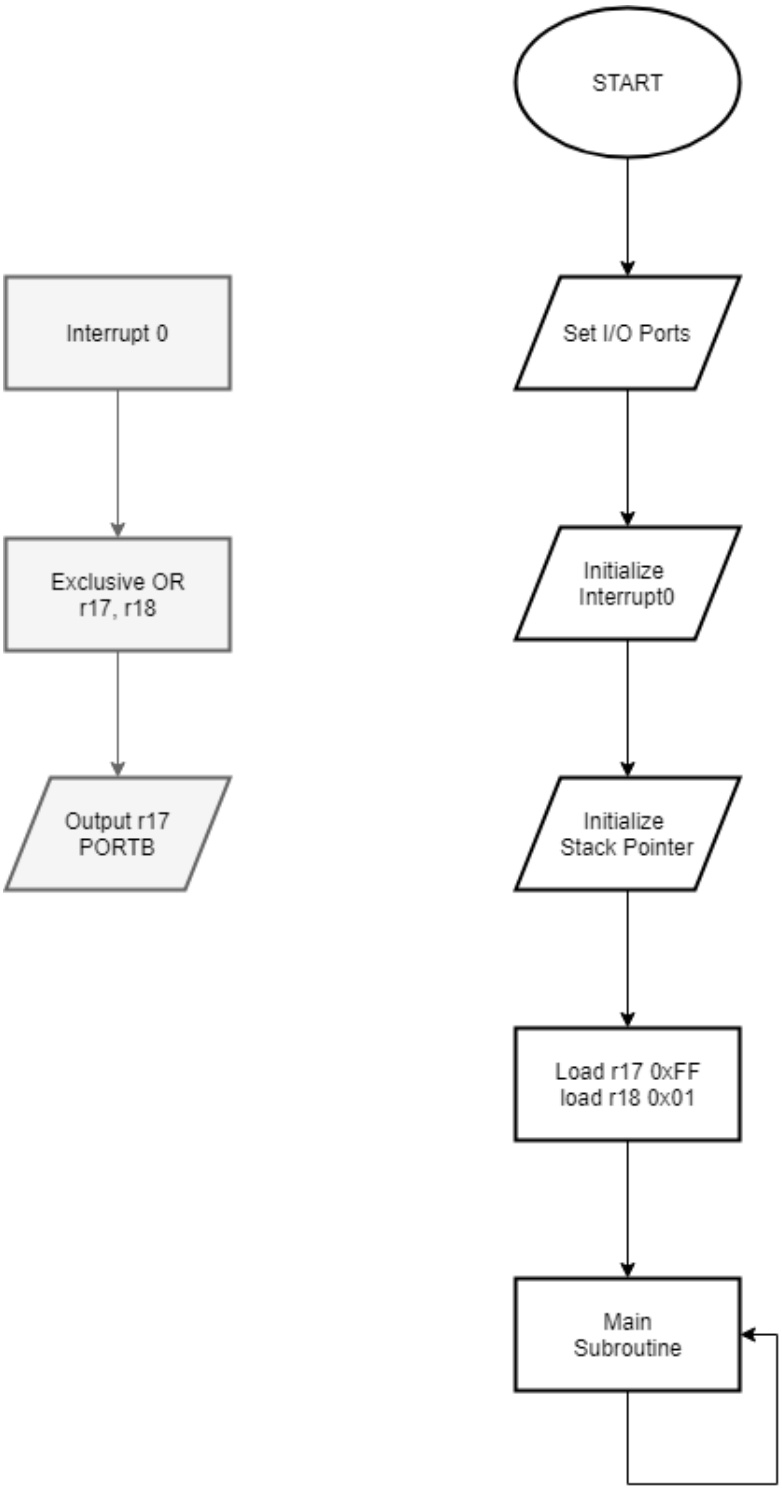

Figure 1: Task 1 diagram

1.2 Code

Listing 1: Task 1 code.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2018-09-24
; Author:
; Jiahui Le (jl224bn)
; Helena Tevar (ht222fd)
;
; Lab number: 3
; Title: Light led with interrupt
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Write a program which turns on
; led 1 with interrupt
;
; Input ports: Port D
;
; Output ports: Port B
;
; Subroutines: delay
; Main
; changeLeds
;
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: 1/10/2018 - "programmed and implemented in board"
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.def leds = r17
.def ledOn = r18

.org 0x00
rjmp start

.org INT0addr
rjmp changeLeds

.org 0x72



;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Start subroutine
;
; Initialize SP, Stack Pointer
; setting I/O ports
```

```
; setting interrupts
; load led values
; Global interrupt enabled
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

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, 0xff
out DDRB, r16

ldi r16, 0x00

out DDRD, r16

ldi r16, 0b00000001
out EIMSK, r16

ldi r16, 0b00000010
sts EICRA, r16


ldi ledOn, 0b00000001
ldi leds, 0b11111111
sei

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;Main loop
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

main:
  nop

rjmp main

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;changeLeds interrupt subroutines
;run when interrupt
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
changeLeds:
            eor leds, ledOn
            out PORTB, leds
reti
```

2 Task 2

Switch – Ring counter / Johnson counter, with interrupt

2.1 Diagram

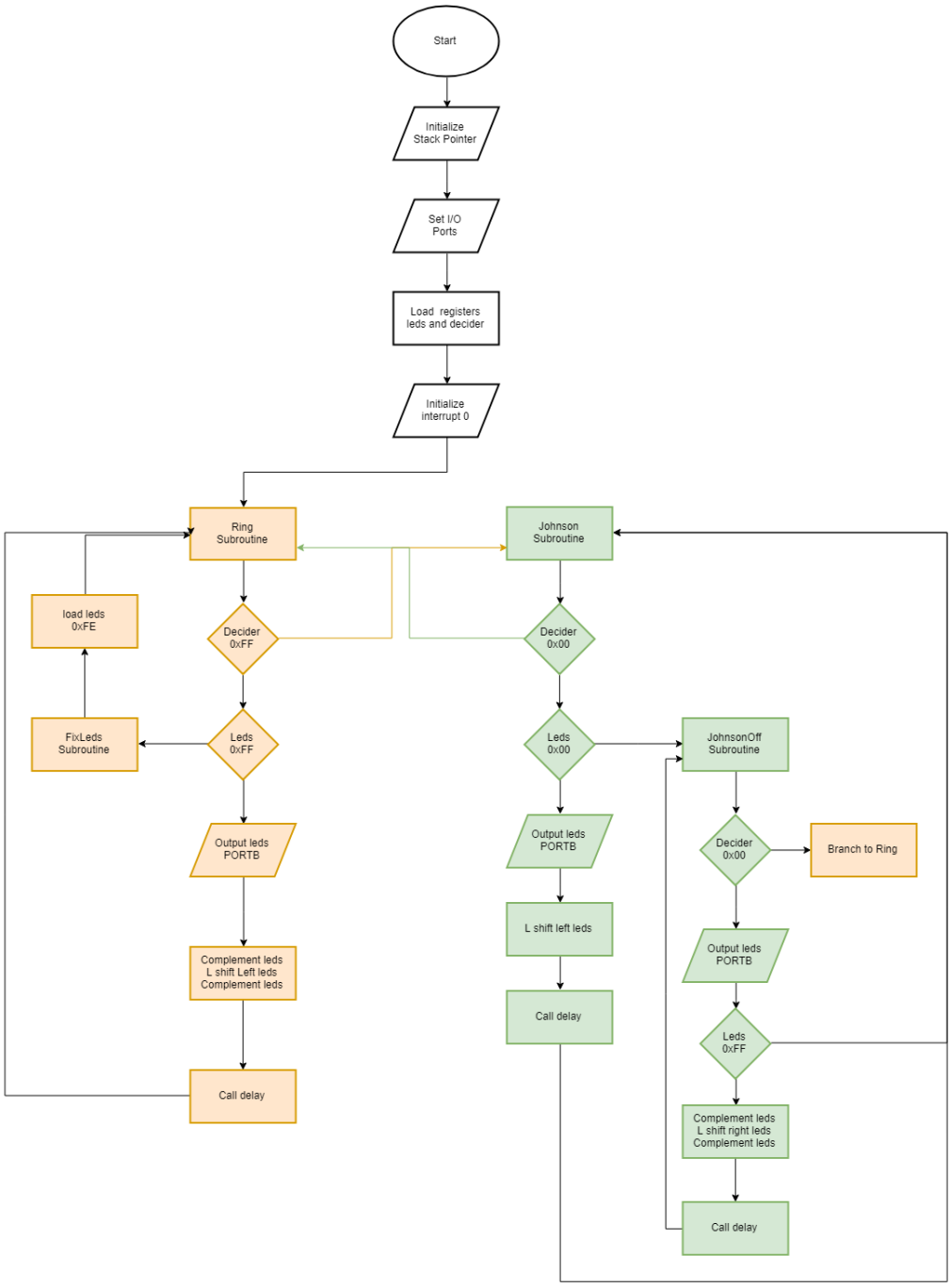

Figure 2: Task 2 diagram

2.2 Code

Listing 2: Task 2 code.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2018-09-24
; Author:
; Jiahui Le (jl224bn)
; Helena Tevar (ht222fd)
;
; Lab number: 3
; Title: Switch - Ring counter / Johnson counter
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Write a program which switches between
; Ring and Johnson counters.
;
; Input ports: Port D
;
; Output ports: Port B
;
; Subroutines: delay
; start
; ring
;           johnson_on
;           johnson_off
;           interrupt_0
;           reset_ring
;           reset_john
;
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: 1/10/2018 - "programmed␣and␣implemented␣in␣board"
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.def leds = r16
.def decider = r22
.def settings = r17

.org 0x00
rjmp start
.org INT0addr
rjmp interrupt_0
.org 0x72
```

```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Start
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

start:

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Start subroutine
;
; Initialize SP, Stack Pointer
; setting I/O ports
; load leds and decider
; setting interrupts
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi r20, HIGH(RAMEND)
out SPH,r20
ldi R20, low(RAMEND)
out SPL,R20

ldi settings, 0xFF
out DDRB, settings
ldi settings, 0x00
out DDRD, settings

ldi leds, 0xFE
ldi decider, 0x00

ldi settings, 0b0000_0001
out EIMSK, settings
ldi settings, 0b0000_0010
sts EICRA, settings

sei



;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; RING COUNTER
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

ring:
        cpi decider,0xFF
        breq reset_john

        cpi leds, 0xFF
        breq fixLedsOff

        out PORTB, leds
        com leds
        lsl leds
        com leds
        rcall delay
```

```
        rjmp ring

        fixLedsOff:
                ldi leds, 0xFE
                rjmp ring


;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;JOHNSON COUNTER
;Fixed without asr
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
johnson_on:
        cpi decider,0x00
        breq 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
        lsr leds
        com leds
        rcall delay
        rjmp johnson_off

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; 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

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; What happens when
; Interrupt 0 pressed
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

interrupt_0:
        com decider
        reti


;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; subroutines to change
; from ring to john
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
reset_ring:
        ldi leds, 0xFF
        out PORTB, leds
        rjmp ring

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

3 Task 3

Rear lights on a car

3.1 Diagram

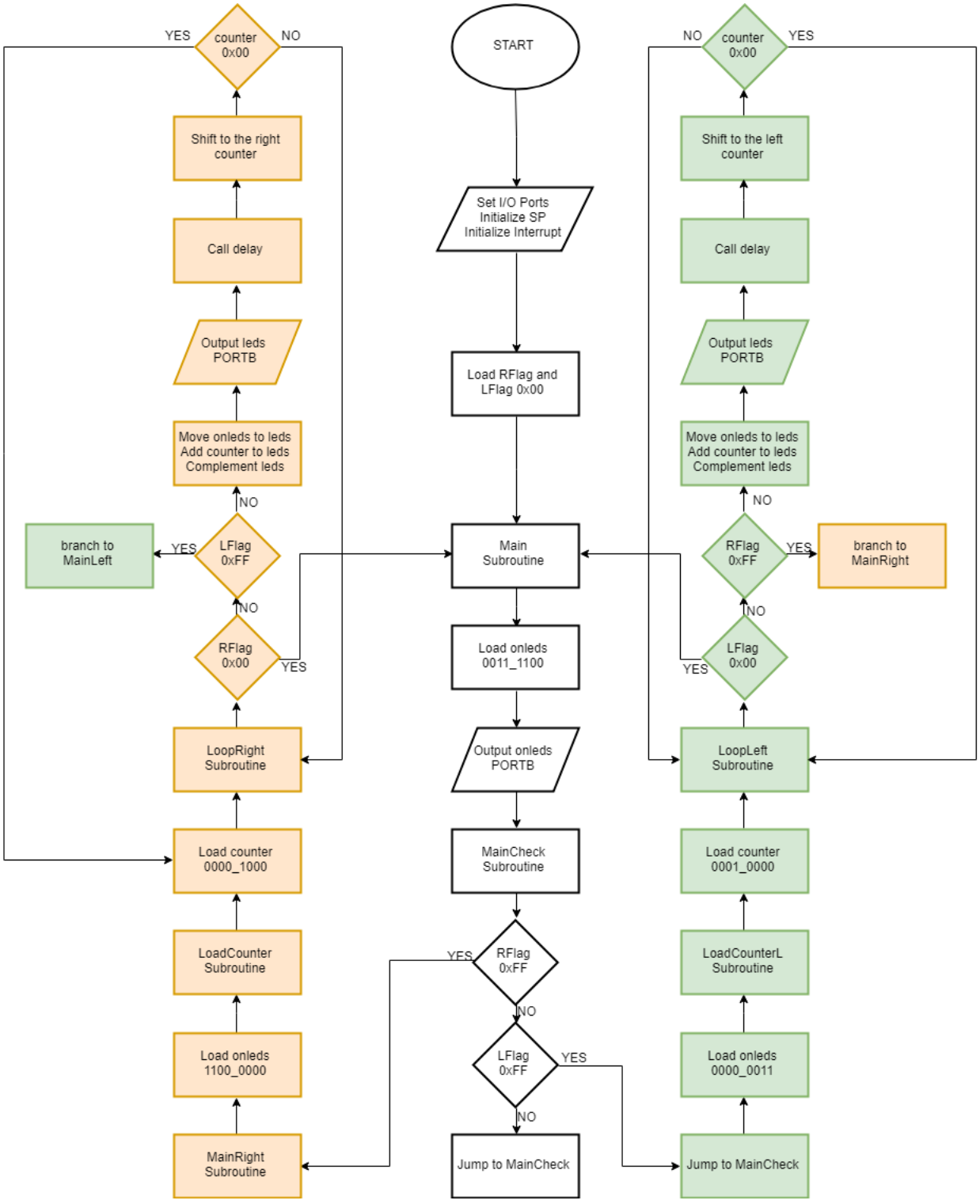

Figure 3: Task 3 Part 1 diagram

Figure 4: Task 3 Part 2 diagram

3.2 Code

Listing 3: Task 3 code.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2018-09-24
; Author:
; Jiahui Le (jl224bn)
; Helena Tevar (ht222fd)
;
; Lab number: 3
; Title: Rear lights on a car
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Simulate the rear lights on a car
;
; Input ports:
;
; Output ports: Port B
;
; Subroutines: START
;           Main - Leds 0011_1100
;           MainLeft - Leds 1110_1100
;           MainRight - Leds 0011_0111
;           Interrupt0
;           Interrupt3
; Included files: m2560def.inc
;
; Other information: The program works using registers as
;           flags and checking their state to
;           change from left to right to Main
;           at any moment. There is no need to go
;           to main to change directions
;
; Changes in program: 1/10/2018 - "programmed and implemented in board"
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.def settings = r16
.def counter = r17
.def leds = r21
.def onleds = r22
.def RFlag = r23
.def LFlag = r24



.org 0x00
rjmp start

.org INT0addr
```

```
rjmp interrupt_0
.org INT3addr
rjmp interrupt_3

.org 0x72

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Start subroutine
;
; Initialize SP, Stack Pointer
; setting I/O ports
; setting interrupts
; load flags values
; Global interrupt enabled
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
start:
ldi settings, 0xFF
out DDRB, settings
ldi RFlag, 0x00
ldi LFlag, 0x00
ldi r20, HIGH(RAMEND)
out SPH,R20
ldi R20, low(RAMEND)
out SPL,R20
ldi settings, 0b0000_1001
out EIMSK, settings
ldi settings, 0b1000_0010
sts EICRA, settings
sei

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Main subroutine
; load leds that should be turned always on
; in this case 0011_1100
; CheckMain:
; Check flags for Changes
; loop if remain unchanged
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Main:
ldi onleds, 0b0011_1100
out PORTB, onleds
;Check interrupts
checkMain:
cpi RFlag, 0xFF
breq MainRight
cpi LFlag, 0xFF
breq MainLeft

rjmp checkMain

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Interrupt 0
```

```
; Change state in right flag
; clear left flag to avoid conflict
; Interrupt 3
; Change state in left flag
; clear right flag to avoid conflict
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;INT0 = RIGHT
interrupt_0:
com RFlag
clr LFLag
reti

;INT3 = LEFT
interrupt_3:
com LFlag
clr RFlag
reti

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
; Main Left loop
; Onleds loaded with the code for the static leds
; Counter loaded with first led to shift to the left
; loopLeft:
; Compare flags state for possible changes
; Move to leds the onleds value and add the value of counter
; Complement leds and output in PORTB
; Shift the counter to the left and loop until it reaches 0x00
; When counter is 0x00, reset counter in loadCounterL
; Loop until flags state changes
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
MainLeft:
        ldi onleds, 0b0000_0011

        loadCounterL:
        ldi counter, 0b0001_0000

        loopLeft:

        cpi LFlag, 0x00
        breq Main
        cpi RFlag, 0xFF
        breq MainRight
        ;add always−on leds to moving leds
        mov leds, onleds
        add leds, counter
        com leds
        ;Showing leds
        out PORTB, leds
        rcall delay
        ;Move leds
        lsl counter
        cpi counter, 0x00
```

```
        breq loadCounterL

        rjmp loopLeft


        ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
        ; Main Right loop
        ; Onleds loaded with the code for the static leds
        ; Counter loaded with first led to shift to the Right
        ; loopLeft:
        ; Compare flags state for possible changes
        ; Move to leds the onleds value and add the value of counter
        ; Complement leds and output in PORTB
        ; Shift the counter to the Right and loop until it reaches 0x00
        ; When counter is 0x00, reset counter in loopright
        ; Loop until flags state changes
        ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

MainRight:
        ldi onleds, 0b1100_0000

        loadCounter:
        ldi counter, 0b0000_1000

        loopright:
        cpi RFlag, 0x00
        breq Main
        cpi LFlag, 0xFF
        breq MainLeft

        mov leds, onleds
        add leds, counter
        com leds

        out PORTB, leds
        rcall delay

        lsr counter
        cpi counter, 0x00
        breq loadCounter

        rjmp loopright

; 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 r20, 86
L1: dec r20
    brne L1
    dec r19
    brne L1
    dec r18
    brne L1
    rjmp PC+1
ret
```

4 Task 4

Rear lights on a car, with light for brakes

4.1 Diagram

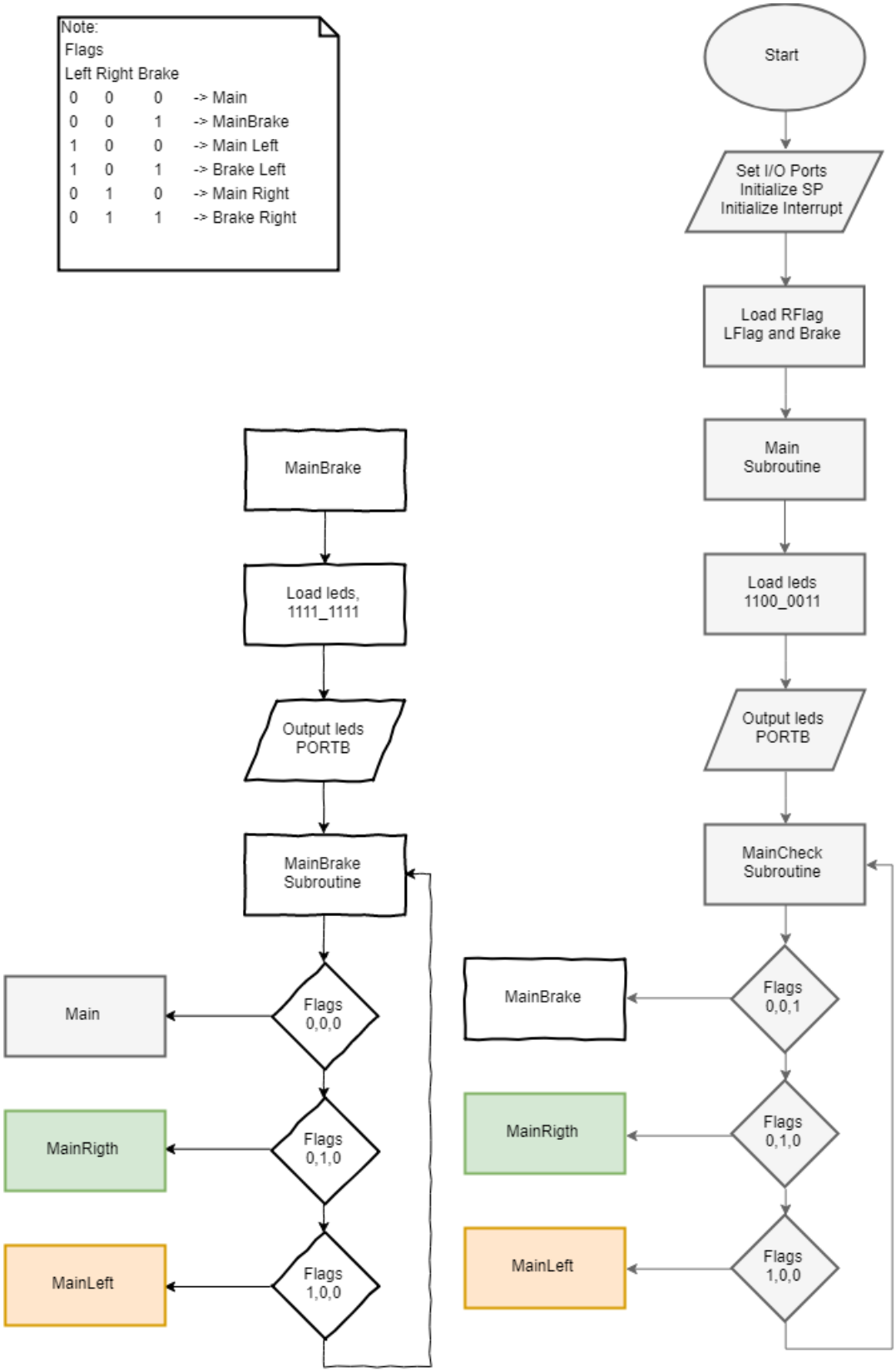

Figure 5: Task 4 Part 1 diagram

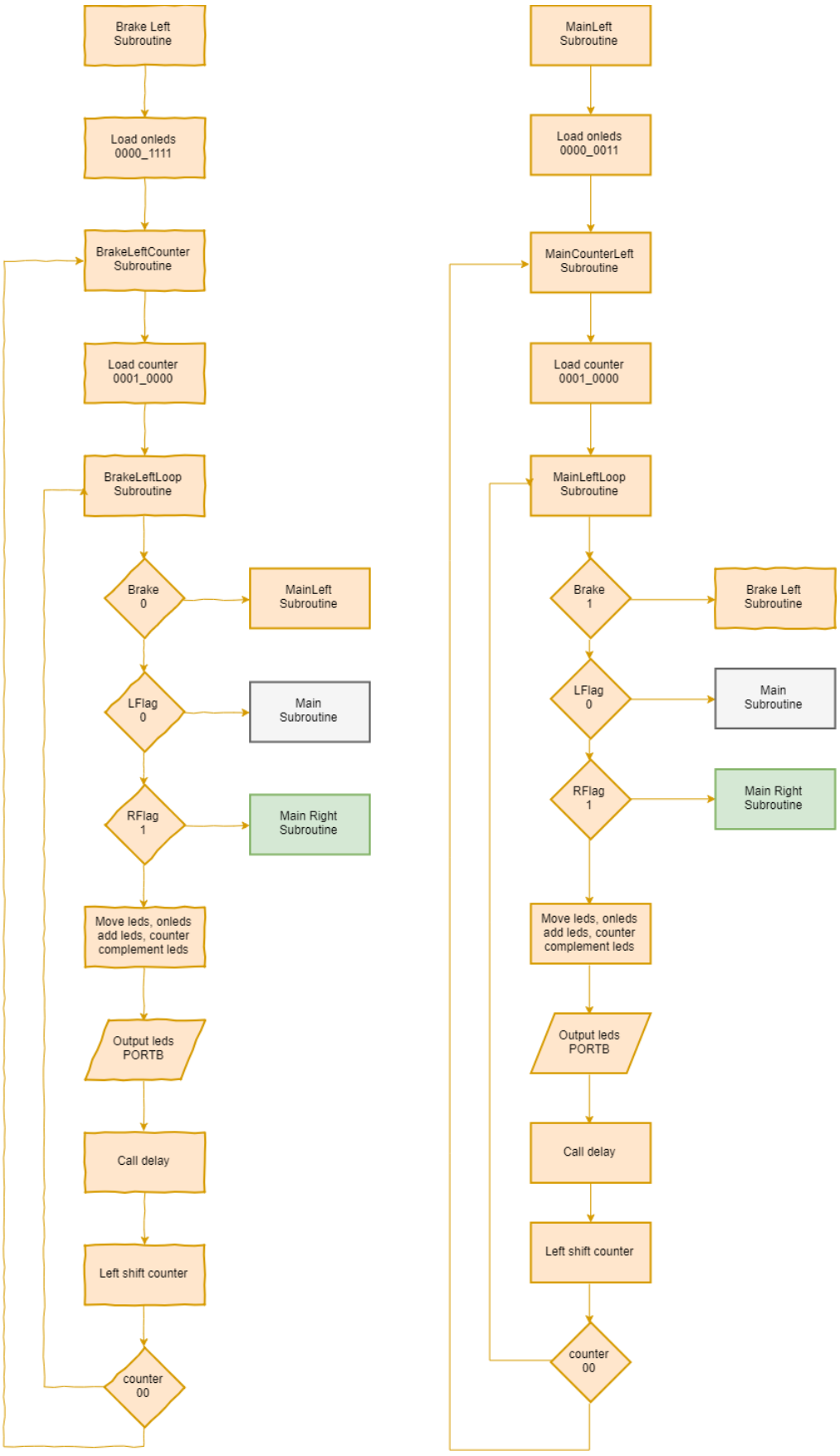

Figure 6: Task 4 Part 2 diagram

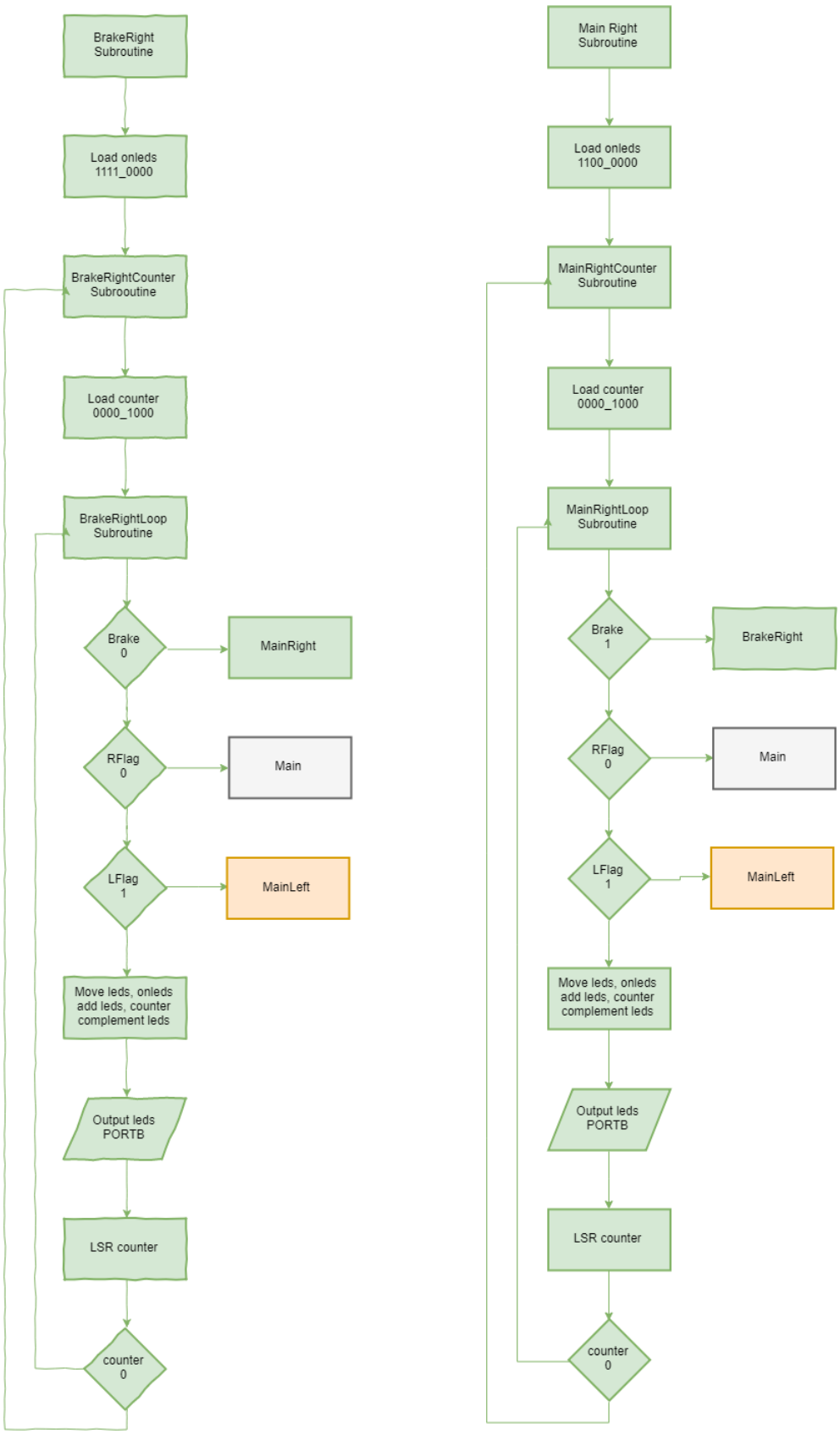

Figure 7: Task 4 Part 3 diagram

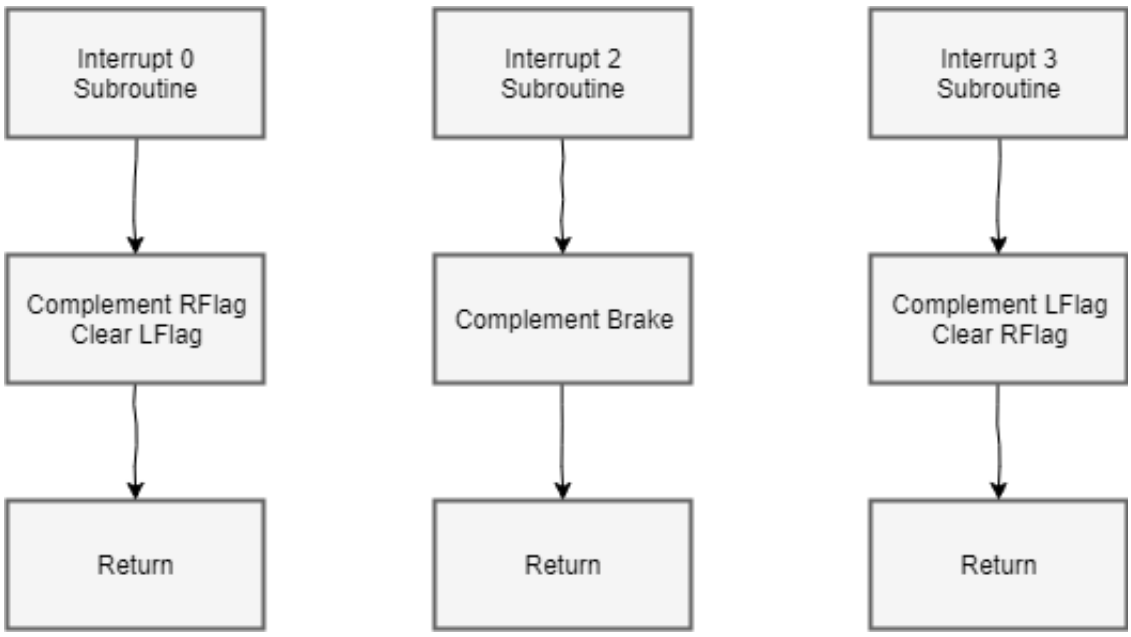

Figure 8: Task 4 Part 4 diagram

4.2 Code

Listing 4: Task 4 code.

;»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»> ; 1DT301, Computer Technology I ; Date: 2018-09-24 ; Author: ; Jiahui Le (jl224bn) ; Helena Tevar (ht222fd) ; ; Lab number: 3 ; Title: Rear lights on a car, ; with light for brakes ; ; Hardware: STK600, CPU ATmega2560 ; ; Function: Changes the leds of the board according ; to the leds of a car, with brake included ; ; Input ports: ; ; Output ports: Port B ; ; Subroutines: ; Main ; MainBrake ; Main Left ; Brake Left ; Main Right ; Brake Right ; ; ; Included files: m2560def.inc ; ; Other information: ; Patter followed for flag states ; Left Right Brake ; 0 0 0 -> Main ; 0 0 1 -> MainBrake ; 1 0 0 -> Main Left ; 1 0 1 -> Brake Left ; 0 1 0 -> Main Right ; 0 1 1 -> Brake Right ; ; Changes in program: 1/10/2018 - "programmed and implemented in board" ; ;«««««««««««««««««««««««««««««««««<

.include "m2560def.inc"

.def settings = r16 .def counter = r17 .def leds = r21 .def onleds = r22 .def RFlag = r23 .def LFlag = r24 .def brake = r25

.org 0x00 rjmp start

.org INT0addr rjmp interrupt$_0 .orgINT3addrrjmpinterrupt_3 .orgINT2addrrjmpinterrupt_2$

.org 0x72 ;«««««««««««««««««««««««««««««««««««< ; Start subroutine ; ; Initialize SP, Stack Pointer ; setting I/O ports ; setting interrupts ; load led values ; Global interrupt enabled ;«««««««««««««««««««««««««««««««««< start:

ldi settings, 0xFF out DDRB, settings

ldi RFlag, 0x00 ldi LFlag, 0x00 ldi brake, 0x00

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

ldi settings, 0b0000$_1 101outEIMSK, settingsldisettings, 0b1010_0 010stsEICRA, settingssei$

;«««««««««««««««««««««««««««««««««< ; Main subroutine ; load leds that should be turned always on ; in this case 0011$_1 100; CheckMain :; Checkflags forChanges; loopifremainunchanged; <<$ $Main :; 0011_1 100ldileds, 0b0011_1 100outPORTB, leds$

MainCheck: cpi RFlag, 0xFF breq MainRight

cpi LFlag, 0xFF breq MainLeft

cpi brake, 0xFF breq MainBrake

rjmp MainCheck

;«««««««««««««««««««««««««««««««««< ; MainBrake subroutine ; load leds that should be turned always on ; in this case 1111$_1 111; MainBrakeCheck :; Checkflags forChanges; loopifremainu$ $MainBrake :; 1111_1 111ldileds, 0x00outPORTB, leds$

MainBrakeCheck: cpi RFlag, 0xFF breq MainRight

cpi LFlag, 0xFF breq MainLeft

cpi Brake, 0x00 breq Main

rjmp MainBrakeCheck

;«««««««««««««««««««««««««««««««« ; Main Left loop ; Onleds loaded with the code for the static leds ; Counter loaded with first led to shift to the left ; loopLeft: ; Compare flags state for possible changes ; Move to leds the onleds value and add the value of counter ; Complement leds and output in PORTB ; Shift the counter to the left and loop until it reaches 0x00 ; When counter is 0x00, reset counter in loadCounterL ; Loop until flags state changes ;««««««««««««««««««««««««««««««<

MainLeft: ldi onleds, $0b0000_0011$

MainLeftCounter: ldi counter, $0b0001_0000$

MainLeftLoop: cpi brake, 0xFF breq BrakeLeft cpi LFlag, 0x00 breq Main cpi RFlag, 0xFF breq MainRight

mov leds, onleds add leds, counter com leds out PORTB, leds rcall delay

lsl counter cpi counter, 0x00 breq MainLeftCounter rjmp MainLeftLoop

;«««««««««««««««««««««««««««««««« ; BrakeLeft loop ; Onleds loaded with the code for the static leds ; Counter loaded with first led to shift to the left ; BrakeLeftLoop: ; Compare flags state for possible changes ; Move to leds the onleds value and add the value of counter ; Complement leds and output in PORTB ; Shift the counter to the left and loop until it reaches 0x00 ; When counter is 0x00, reset counter in loadCounterL ; Loop until flags state changes ;««««««««««««««««««««««««««««««« BrakeLeft: ldi onleds, $0b0000_1111$

BrakeLeftCounter: ldi counter, $0b0001_0000$

BrakeLeftLoop: cpi brake, 0x00 breq MainLeft cpi RFlag, 0xFF breq MainRight cpi LFlag, 0x00 breq Main

mov leds, onleds add leds, counter com leds out PORTB, leds rcall delay

lsl counter cpi counter, 0x00 breq BrakeLeftCounter rjmp BrakeLeftLoop

;«««««««««««««««««««««««««««««««« ; Main Right loop ; Onleds loaded with the code for the static leds ; Counter loaded with first led to shift to the Right ; loopLeft: ; Compare flags state for possible changes ; Move to leds the onleds value and add the value of counter ; Complement leds and output in PORTB ; Shift the counter to the Right and loop until it reaches 0x00 ; When counter is 0x00, reset counter in loopright ; Loop until flags state changes ;««««««««««««««««««««««««««««««<

MainRight: ldi onleds, $0b1100_0000$

MainRightCounter: ldi counter, $0b0000_1000$

MainRightLoop: cpi brake, 0xFF breq BrakeRight cpi RFlag, 0x00 breq Main cpi LFlag, 0xFF breq MainLeft

mov leds, onleds add leds, counter com leds out PORTB, leds rcall delay

lsr counter cpi counter, 0x00 breq MainRightCounter rjmp MainRightLoop

;«««««««««««««««««««««««««««««««« ; BrakeRight loop ; Onleds loaded with the code for the static leds ; Counter loaded with first led to shift to the Right ; BrakeRightLoop: ; Compare flags state for possible changes ; Move to leds the onleds value and add the

value of counter ; Complement leds and output in PORTB ; Shift the counter to the Right and loop until it reaches 0x00 ; When counter is 0x00, reset counter in loopright ; Loop until flags state changes ;«««««««««««««««««««««««««««««««< BrakeRight: ldi onleds, $0b1111_0000$

BrakeRightCounter: ldi counter, $0b0000_1000$

BrakeRightLoop:

cpi brake, 0x00 breq MainRight cpi RFlag, 0x00 breq GoToMain cpi LFlag, 0xFF breq MainLeft

mov leds, onleds add leds, counter com leds out PORTB, leds rcall delay

lsr counter cpi counter, 0x00 breq BrakeRightCounter rjmp BrakeRightLoop

;«««««««««««««««««««««««««««««««< ; Branch can not jump that far ; but rjmp can. Fixed branch calling a subroutine instead ;«««««««««««««««««««««««««««««««< GoToMain: 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 r20, 86 L1: dec r20 brne L1 dec r19 brne L1 dec r18 brne L1 rjmp PC+1 ret

;«««««««««««««««««««««««««««««««< ; Interrupt 0 ; Change state in right flag ; clear left flag to avoid conflict ; Interrupt 3 ; Change state in left flag ; clear right flag to avoid conflict ; Interrupt 2 ; Complement the state in brake flag ; Doesn't need clear, because you can brake while turning ;«««««««««««««««««««««««««««««««<

interrupt$_0$: $comRFlagclrLFLagreti$

interrupt$_3$: $comLFlagclrRFlagreti$

interrupt$_2$: $combrakereti$