1DT301 lab3

Songho Lee Sarpreet Singh Buttar

September 22, 2016

1 Introduction

This report provides the solutions for the third laboratory of the course 1DT301, which is focusing on practising interrupts. In order to set up an interrupt sequence for switch 0, we therefore connected the switch inputs into portD where it has connection to INT pins of ATMega2560 CPU.

2 Task 1

Task 1 is to write a programme which turns on the LED if it was off, and turn off them if it was already on, by listening to the switch input $SW0$. From this laboratory we are using interrupt to initiate sub-programme inside the task file.

As an initial condition, we are matched our label interrupt programme 'changeLED' to the INT address 0, and we set 1 in External Interrupt Control Register A(EICRA) for ISC00 so that it responds upon switch number 0. Besides, by setting 01 in External Interrupt Mask Register we set any logical change of INT0 generates an interrupt request. Source code is presented below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-19
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 3
;     Title: Task 1
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Switching LEDS by responding to switches
;
;     Input ports: On-board switches on portD
;
;     Output ports: On-board LEDs on PORTB.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;
;     Other information:
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

;Load pre-configured files for the ports, and memory adresses.
.include "m2560def.inc"

.org 0x00
```

```
rjmp start

.org INT0addr
rjmp changeLED

.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

.def LEDSTATUS = R18

ldi r16, 0xFF  ;Set data direction registers.
out DDRB, r16  ;Set B port as output ports
out portB, r16
mov LEDSTATUS, r16

ldi r16, 0x00  ;Set data direction registers.
out DDRD, r16  ;Set D port as input ports

ldi r16, 0b00000010
sts EICRA, r16

ldi r16, 0b00000001
out EIMSK, r16


sei  ;Enable global interrupt

ldi r19,0

main_doing_nothing: nop

rjmp main_doing_nothing


changeLED:
COM LEDSTATUS
out portB, LEDSTATUS
reti
```

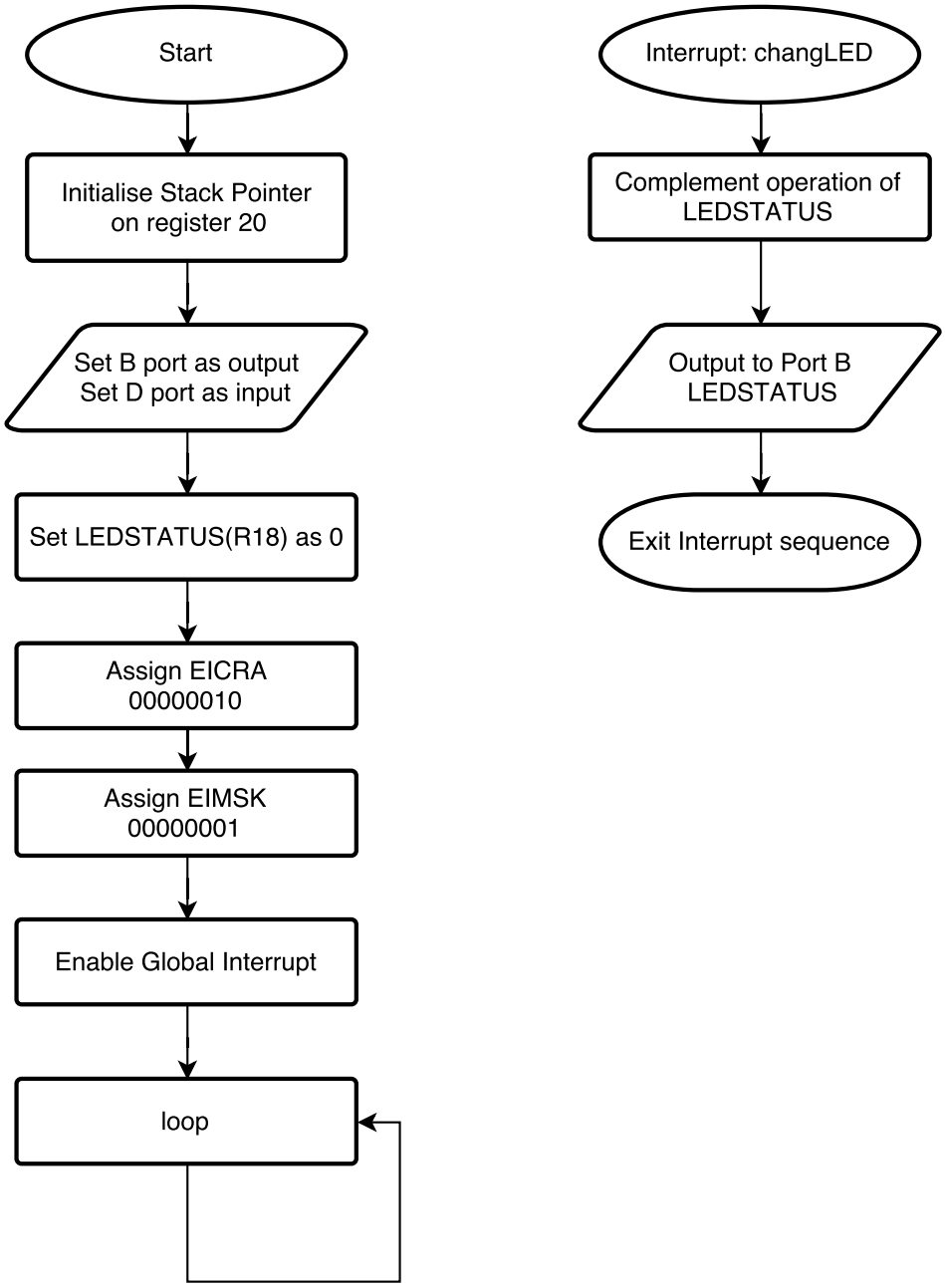

Figure 1: Flowchart of task 1

3 Task 2

Task 2 is about flashing the 8 LEDs either in the form of a ring counter or in the form of a Johnson counter by using the switch SW0 which is connected to PORTD in order to switch between the two counters.
We assigned 0b00000010 to EICRA because we only want to enable SW1. In addition, EIMSK have 0b00000001 value because we want to control any logical change on SW1 generated during an interrupt request. Furthermore, We also used two other registers named as 'COUNTERFLAG' and 'DIRECTIONFLAG'. Counterflag helps to keep track of which counter is running so that next time we will run the opposite counter. Directionflag helps to keep track of going from left to right. Also, it helps in the interrupt to start the next counter from beginning. Note that we have not describe delay in this flow chart because we used the same delay as LAB 2. Mechanism for delay can be found in a flowchart in the second report, if further explanation is required.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-19
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 3
;     Title: Task 2
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Switch counters by responding to switches
;
;     Input ports: On-board switches on portD
;
;     Output ports: On-board LEDs on PORTB.
;
;     Subroutines: Delay
;     Included files: m2560def.inc
;
;     Other information:
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

;Load pre-configured files for the ports, and memory adresses.
.include "m2560def.inc"

.org 0x00
rjmp start

.org INT0addr
rjmp changeCNT

.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

.def N = R18
```

```
.def K = R19

ldi r16, 0xFF  ;Set data direction registers.
out DDRB, r16  ;Set B port as output ports
out portB, r16

ldi r16, 0x00  ;Set data direction registers.
out DDRD, r16  ;Set D port as input ports

ldi r16, 0b00000010
sts EICRA, r16

ldi r16, 0b00000001
out EIMSK, r16


sei  ;Enable global interrupt

ldi r17,0b00000001
mov r21, r17   ;Registering 00000001 on both register 17, and 21.
ldi r25,0b11111111
ldi r22,0
.def DIRECTIONFLAG = R22
ldi r23,0
.def COUNTERFLAG = R23

mainloop:

firstloop:

rcall overalldelay

;COMPLEMNENT FOR LED
mov r16, r17
com r16
out portB, r16
;LED OPERATION DONE

cpi COUNTERFLAG, 0
brne johnson
ring:
LSL r17
cpi COUNTERFLAG, 0
breq johnsonend

johnson:

cpi DIRECTIONFLAG,0xFF ;RIGHT
breq shiftright

;COMPARISION
shiftleft:

LSL r17
add r17, r21

cpi DIRECTIONFLAG,0xFF
brne begincompare

shiftright:
```

```
LSR r17

begincompare:
cp r17, r25
breq equal

johnsonend:


cpi COUNTERFLAG,0
brne ringendconditiondone

cpi r17, 0
brne ringendconditiondone
ldi r17, 1

ringendconditiondone:

rjmp firstloop

equal:
com r25
com DIRECTIONFLAG

rjmp mainloop


;Interrupt

changeCNT:
com COUNTERFLAG
ldi R17, 0b00000001

reti


;Subroutines.

;;DELAY
overalldelay:
push N
push K
ldi N, 50;      r18 to be our N
ldi K, 0;       r19 to be our counter for all steps
superdelay1:
     push K
     ldi K, 0;
     outerdelay1:
     push K
     ldi K, 0;

     innerdelay1:
     inc K
     cp N, K
     brge innerdelay1
     pop K

     inc K
     cp N, K
     brge outerdelay1
```

```
    pop K

    inc K
    cp N, K
    brge superdelay1

pop K
pop N

ret
;;DELAY ENDS
```

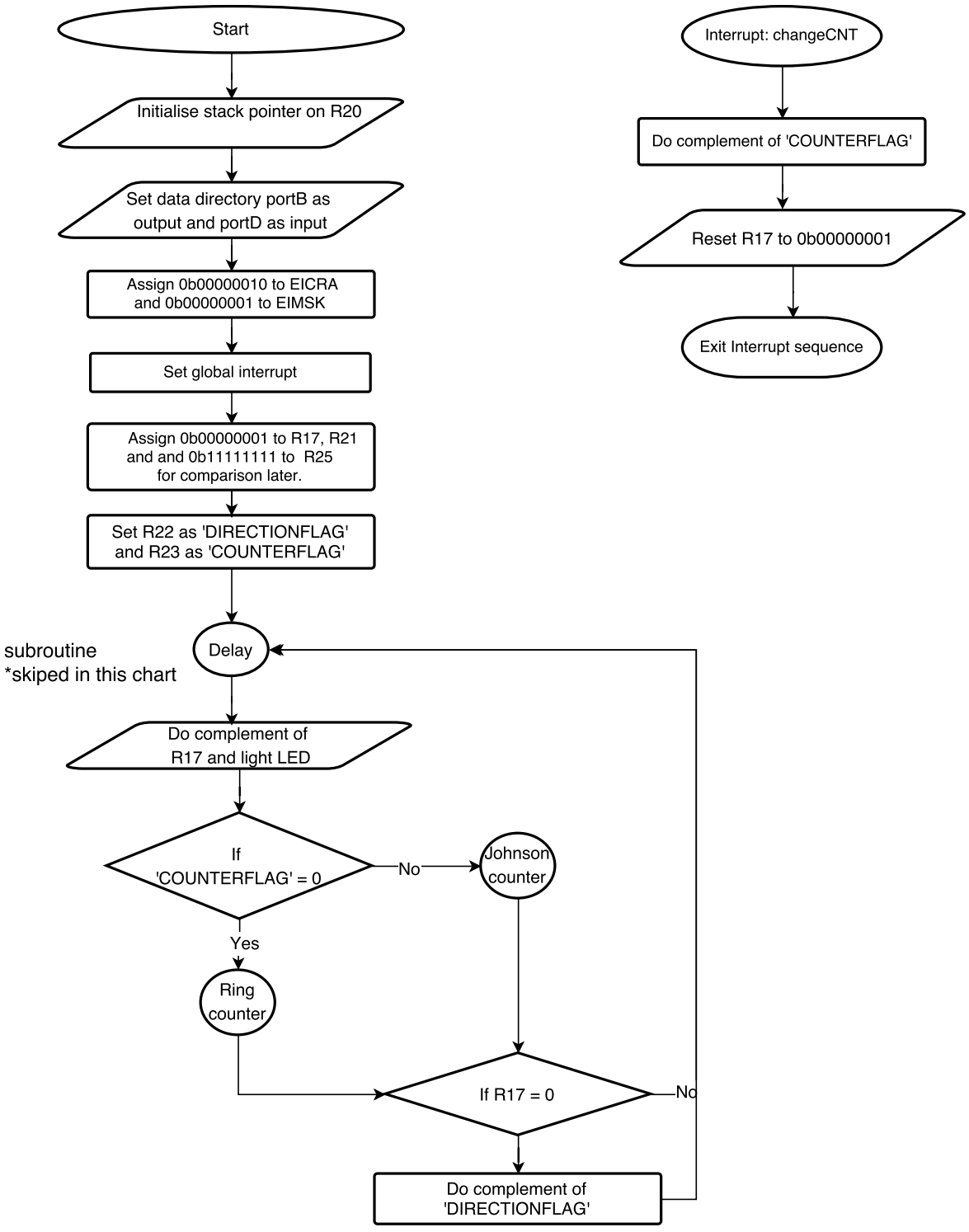

Figure 2: Flowchart of task 2

4 Task 3

The third task is to stimulate rear lights on a car, especially to implement the rear lights for normal operations, and sending a signal(blinking) when a vehicle tries to turn left or right side. We set up SW1 to signal the right-turn, and SW3 to signal the left-turn, in a similar way of previous tasks: setting up ones on External Interrupt Control Register A for SW1,SW3, and marking External Interrupt Mask Register when to initiate interruption. For task three and four we are reacting on a falling edge since it is important in security aspect that it reacts right upon. Note that we used SW1 for signing left turn, instead of SW0 due to failure of the switch on the board we used.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-19
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 3
;     Title: Task 3
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Switch counters by responding to switches
;
;     Input ports: On-board switches on portD
;
;     Output ports: On-board LEDs on PORTB.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;
;     Other information:
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00
rjmp start

.org INT1addr
rjmp blinkRIGHT
.org INT3addr
rjmp blinkLEFT

.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

.def N = R18
.def K = R19

ldi r16, 0xFF  ;Set data direction registers.
out DDRB, r16  ;Set B port as output ports
```

```
out portB, r16

ldi r16, 0x00  ;Set data direction registers.
out DDRD, r16  ;Set D port as input ports


ldi r16, 0b10001000 ;External interrupt control register A
sts EICRA, r16

ldi r16, 0b00001010 ;Which interrupts to enable
out EIMSK, r16


sei  ;Enable global interrupt


.def normal_left = R22
ldi normal_left, 0b11000000
.def normal_right = R23
ldi normal_right, 0b00000011


mainloop:
ldi R17,0
add R17, normal_left
add R17, normal_right

rcall putLED

rjmp mainloop


;Interrupt

blinkLEFT:

push r17
push r24

ldi r24,0
setleft:
ldi R21, 0b00010000
mov R17, R21
add R17, normal_right

leftloop:
rcall putLED
rcall overalldelay

cpi R21, 0
breq setleft

lsl R21
mov R17, R21
add R17, normal_right

in R24, PIND
cpi R24, 0b11110111
breq quitleft

```

```
rjmp leftloop
quitleft:
pop r24
pop r17

reti

blinkRIGHT:
push r17
push r24

setright:

ldi R21, 0b00001000
mov R17, R21
add R17, normal_left
rightloop:

rcall putLED
rcall overalldelay

cpi R21, 0
breq setright

lsr R21
mov R17, R21
add R17, normal_left

in r24, pind
cpi r24, 0b11111101
breq quitright

rjmp rightloop

quitright:
pop r24
pop r17
reti

;Subroutines.

;;DELAY
overalldelay:
push N
push K
ldi N, 60;      r18 to be our N
ldi K, 0;       r19 to be our counter for all steps
superdelay:
     push K
     ldi K, 0;
     outerdelay:
     push K
     ldi K, 0;

     innerdelay:
     inc K
     cp N, K
     brge innerdelay
     pop K
```

```
        inc K
        cp N, K
        brge outerdelay
        pop K

        inc K
        cp N, K
        brge superdelay

pop K
pop N

ret

putLED:
;COMPLEMNENT FOR LED
push r16
mov r16, r17
com r16
out portB, r16
;LED OPERATION DONE
pop r16
ret
;;DELAY ENDS
```

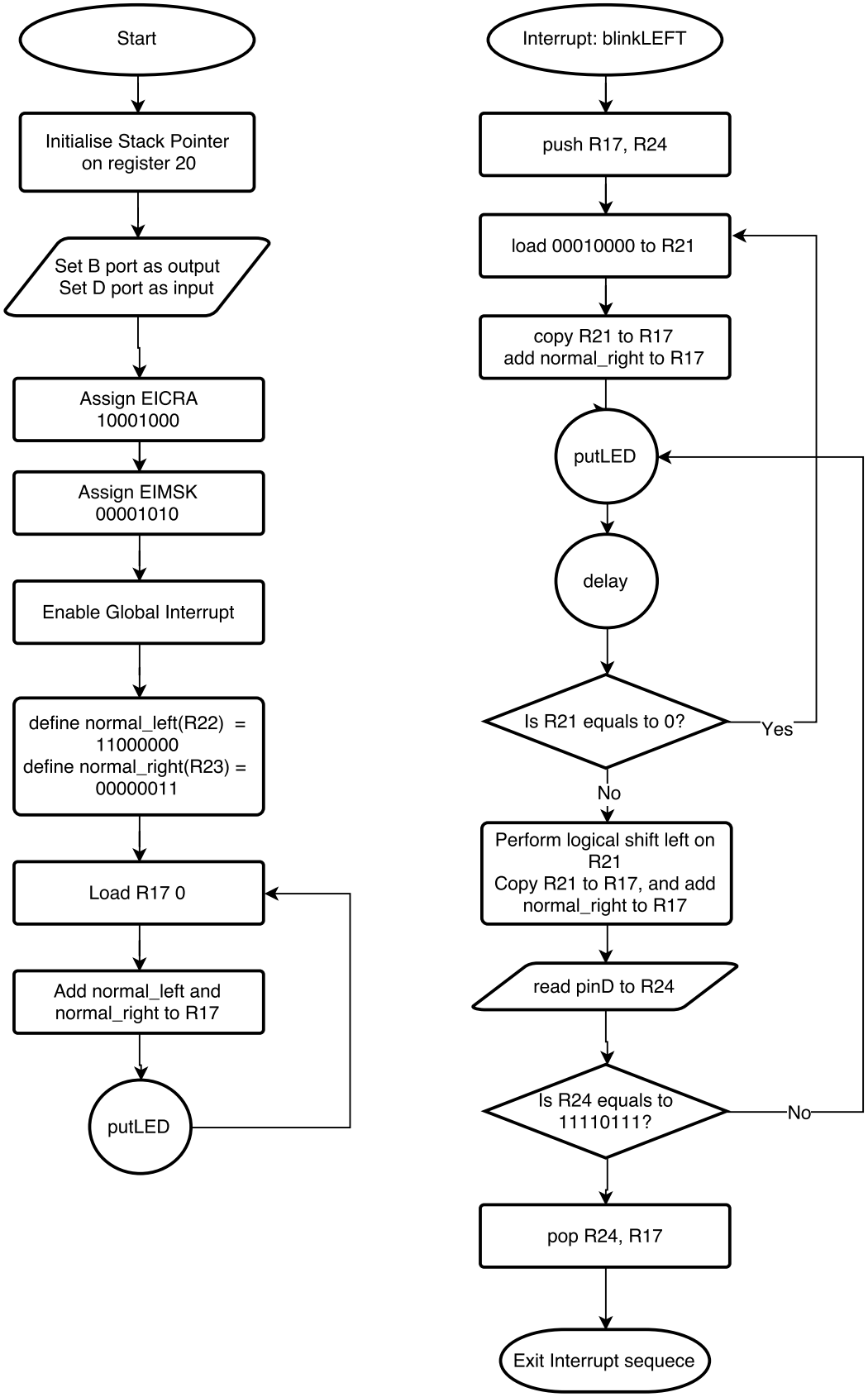

Figure 3: Flowchart of task 3

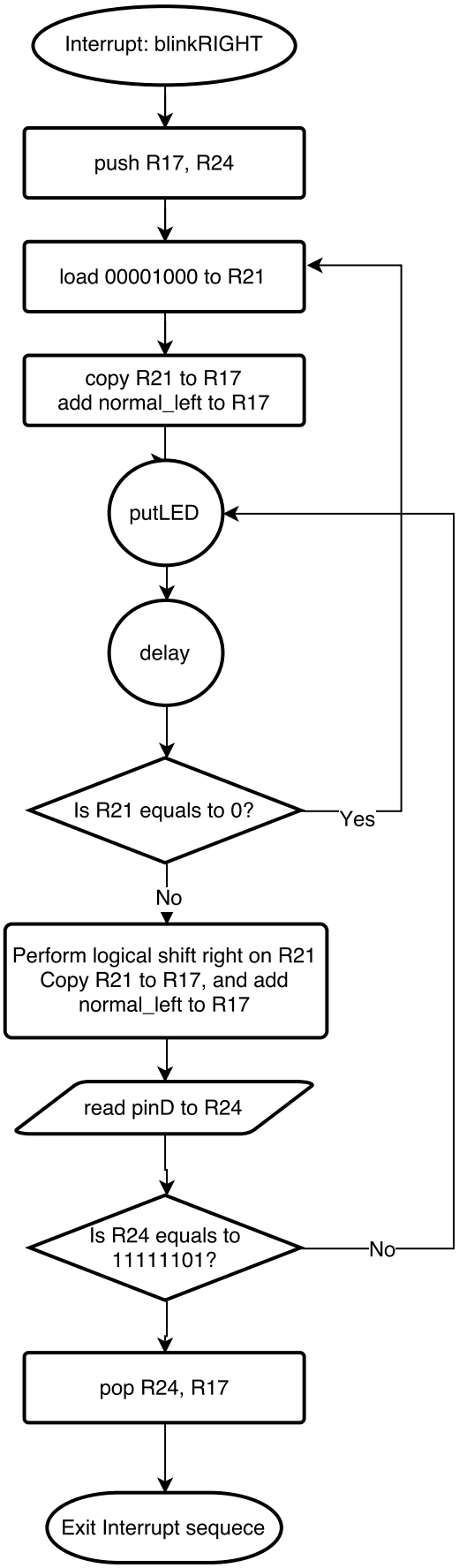

Figure 4: Flowchart of task 3-1

5 Task 4

Task 4 is the extended version of Task 3. We added INT2 for SW2 for the 'BREAK' function. In this task, we are also enabling global interrupts while turning left and right so that if 'BREAK' function is on we can handle the turning and break LEDs at the same time.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-19
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 3
;     Title: Task 4
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Switch counters by responding to switches
;
;     Input ports: On-board switches on portD
;
;     Output ports: On-board LEDs on PORTB.
;
;     Subroutines: If applicable.
;     Included files: m2560def.inc
;
;     Other information:
;     Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00
rjmp start

.org INT1addr
rjmp blinkRIGHT
.org INT2addr
rjmp blinkBREAK
.org INT3addr
rjmp blinkLEFT

.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

.def N = R18
.def K = R19

ldi r16, 0xFF  ;Set data direction registers.
out DDRB, r16  ;Set B port as output ports
out portB, r16

ldi r16, 0x00  ;Set data direction registers.
```

```
out DDRD, r16  ;Set D port as input ports

;External interrupt control register A
ldi r16, 0b10101000 ;INT3, INT2, INT1 on falling edge
sts EICRA, r16

ldi r16, 0b00001110 ;Which interrupts to enable
out EIMSK, r16


sei  ;Enable global interrupt


.def normal_left = R22
ldi normal_left, 0b11000000
.def normal_right = R23
ldi normal_right, 0b00000011


mainloop:
ldi R17,0
add R17, normal_left
add R17, normal_right

rcall putLED

rjmp mainloop


;Interrupt

blinkLEFT:
;rcall overalldelay

cpi R25,1
breq quitleft

ldi R25,1

push r17
push r24

sei

ldi r24,0
setleft:
ldi R21, 0b00010000
mov R17, R21
add R17, normal_right

leftloop:
rcall putLED
rcall overalldelay

cpi R21, 0
breq setleft

lsl R21
mov R17, R21
add R17, normal_right
```

```
cpi R25,0
breq quitleft
rjmp leftloop

quitleft:
pop r24
pop r17
ldi R25,0
reti


blinkRIGHT:
;rcall overalldelay
cpi R25,2
breq quitright

ldi R25,2

push r17
push r24

sei

setright:

ldi R21, 0b00001000
mov R17, R21
add R17, normal_left
rightloop:

rcall putLED
rcall overalldelay

cpi R21, 0
breq setright

lsr R21
mov R17, R21
add R17, normal_left

;in r24, pind
;cpi r24, 0b11111101
;breq quitright

cpi R25,0
breq quitright

rjmp rightloop

quitright:
pop r24
pop r17
ldi R25,0

reti

blinkBREAK:
ldi normal_left, 0b11110000
ldi normal_right, 0b00001111
```

```
cpi r25,2
breq setright2
cpi r25,0
breq breakloop

setleft2:
ldi R21, 0b00010000
mov R17, R21
add R17, normal_right

leftloop2:
rcall putLED
rcall overalldelay

cpi R21, 0
breq setleft2

lsl R21
mov R17, R21
add R17, normal_right

in r24, pind
cpi r24, 0b11111111
breq resetbreak

rjmp leftloop2

setright2:

ldi R21, 0b00001000
mov R17, R21
add R17, normal_left
rightloop2:

rcall putLED
rcall overalldelay

cpi R21, 0
breq setright2

lsr R21
mov R17, R21
add R17, normal_left

in r24, pind
cpi r24, 0b11111111
breq resetbreak

rjmp rightloop2

breakloop:
ldi r17,0

add r17,normal_left
add r17,normal_right

rcall putlED

in r24, pind
```

```
    cpi r24, 0b11111111
    breq resetbreak
    rjmp breakloop

resetbreak:
    ldi normal_left, 0b11000000
    ldi normal_right, 0b00000011
    reti

;Subroutines.

;;DELAY
overalldelay:
    push N
    push K
    ldi N, 50;      r18 to be our N
    ldi K, 0;       r19 to be our counter for all steps
superdelay:
        push K
        ldi K, 0;
        outerdelay:
        push K
        ldi K, 0;

        innerdelay:
        inc K
        cp N, K
        brge innerdelay
        pop K

        inc K
        cp N, K
        brge outerdelay
        pop K

        inc K
        cp N, K
        brge superdelay

    pop K
    pop N

    ret

putLED:
    ;COMPLEMNENT FOR LED
    push r16
    mov r16, r17
    com r16
    out portB, r16
    ;LED OPERATION DONE
    pop r16
    ret
    ;;DELAY ENDS
```

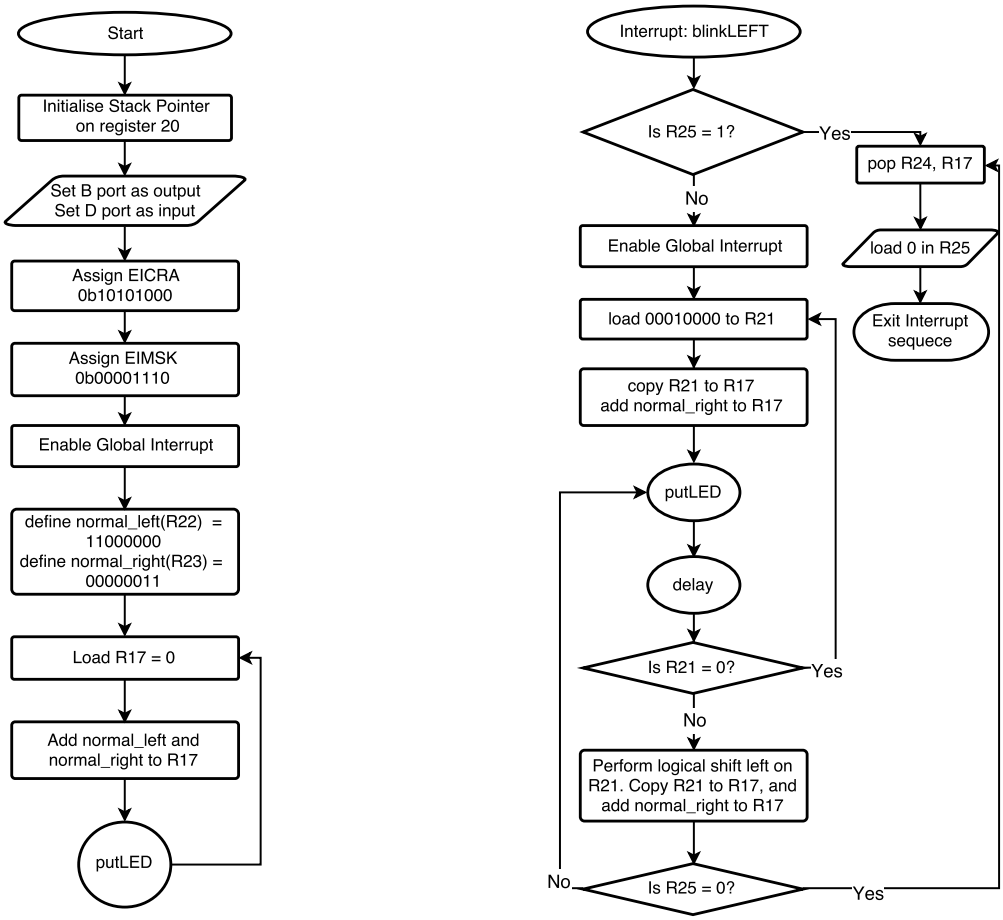

Figure 5: Flowchart of task 4

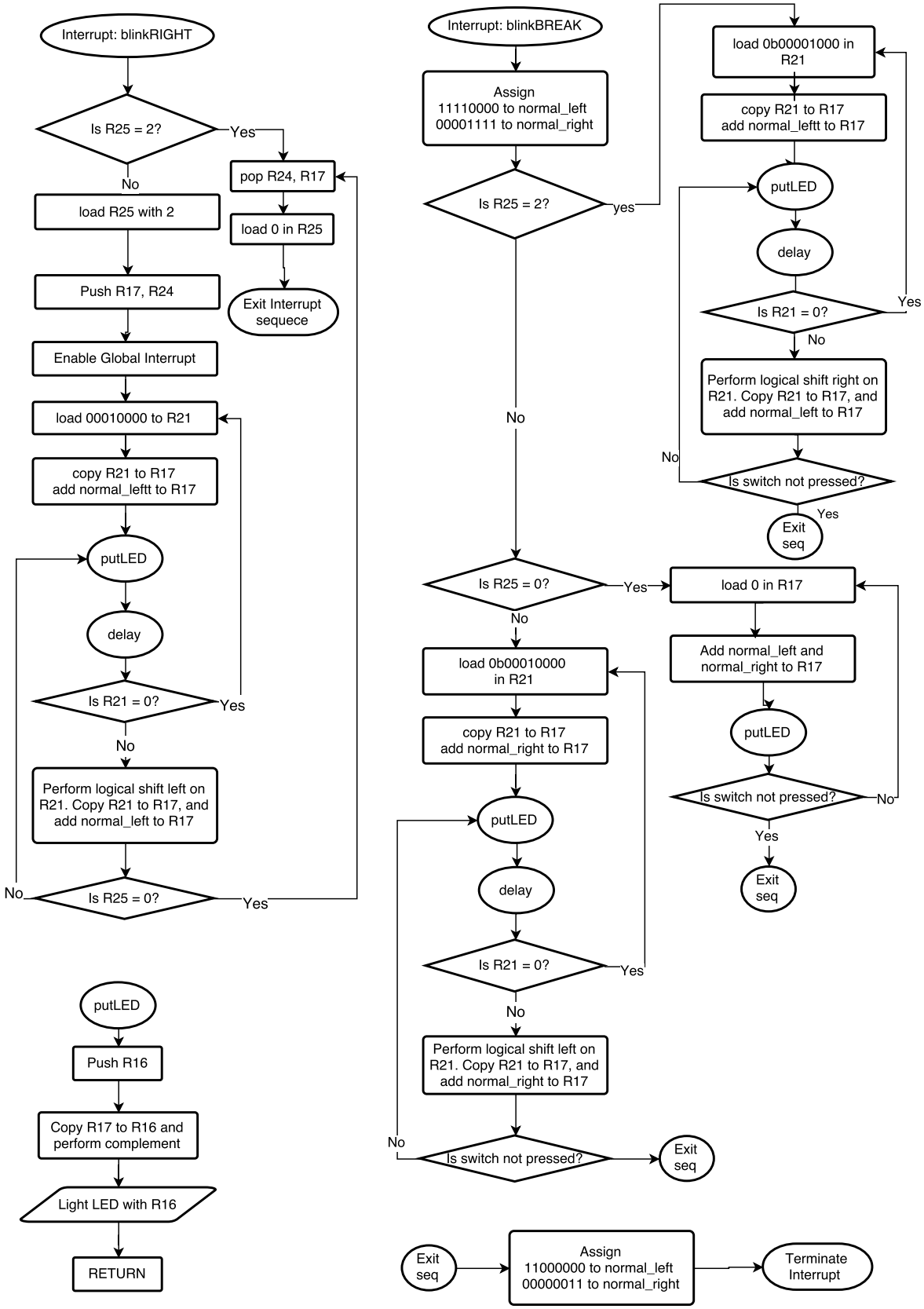

Figure 6: Flowchart of task 4-1