1DT301 lab4

Songho Lee Sarpreet Singh Buttar

October 7, 2016

1 Introduction

This report provides the solutions for the fourth laboratory of the course 1DT301, which is focusing on applying timer(a sort of interrupt) and serial communication on the board STK600 with ATMega2560 CPU. For a timer functionality, 8-bit timer/counter0 is used with purse width modulator(PWM), and for serial input we are using Universal Synchronous and Asynchronous serial Receiver and Transmitter(USART).

2 Task 1

Task 1 is to write an assembly programme which turns on and off the LED with the frequency 1Hz. Duty cycle 50%. (On: 0.5 sec, Off: 0.5 sec.) We understood the mechanism of a timer interrupt as there is a timer/counter which increases on every specific interval(cycles), which is dependent on prescaler, and the interrupt is triggered whenever the 8bit timer/counter(TCNT) overflows. Considering the above mentioned mechanism of a timer interrupt, the oscillation frequency of the CPU is configured to be 1MHz, and a prescaler value for the timer/counter is set up by 1024, which seems to be a maximum value of the current CPU, by allocating the value 101 to Timer/Counter Control Register B(TCCR0B), since the Clock Select (CS02:0) bits inside TCCR0B are controlling clocks for counter. (See 16.3 in doc2549)

However, even if we chose the largest possible prescaler, which is dividing upto 1024 cycles, increasing one timer counter only spends 0.001024 second, and this is yet a far away from our desired value: 0.5 second. Besides increasing TCNT to its maximum value, which is 8 bit, is not still enough Therefore, we decided to introduce another counter that we named CustomCNT, and it is going to increased by one whenever TCNT increased 50 times and calls timer interrupt due to its overflow.

In short, it is designed to spend $1024 * 50$ cycles from timer interrupt side, and 10 times more on our CustomCNT. With this result, it is going to trigger changing LED lights every 0.512 second. The code for the programme is presented below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-26
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 4
;     Title: Task 1
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: generate Pulse with modulation(PWM)
;
;     Input ports: None
;
;     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 initialise

.org OVF0addr
rjmp timer0int

.def temp = r16
.def LEDstat = r17
.def CustomCNT = r18
.equ COUNTVALUE = 206

.org 0x72
initialise:


;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

ldi temp, 0x01 ;Set data direction registers.
out DDRB, temp ;Set B port as output ports
ldi LEDstat, 0x00
out portB, LEDstat


ldi temp, 0b101     ;Setting up a prescaler. See table 42
out TCCR0B, temp
ldi temp, (1<<TOIE0)
sts TIMSK0, temp

ldi temp, COUNTVALUE
out TCNT0, temp

sei  ;Enable global interrupt

ldi CustomCNT,0

start:    ;The relative jump uses two cycles.
rjmp start

timer0int:
push temp
in temp, SREG
push temp

;Reset counter value
ldi temp, COUNTVALUE
```

```
out TCNT0, temp
inc CustomCNT

cpi CustomCNT,10
brne continue

ldi CustomCNT,0
COM LEDstat
out portB, LEDstat

continue: nop
pop temp
out SREG, temp
pop temp
reti
```

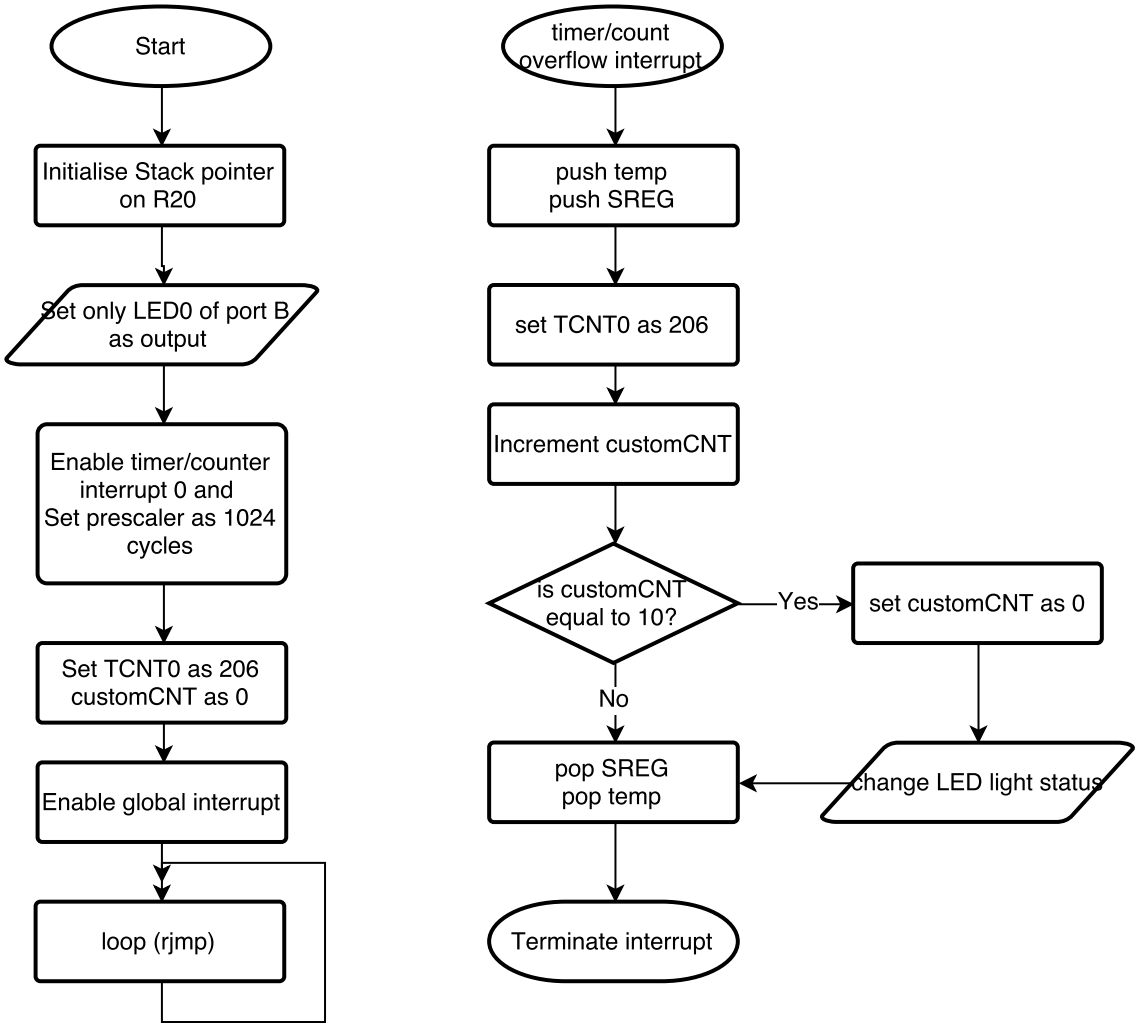

Figure 1: Flowchart of task 1

3 Task 2

Task 2 asks to modify the previous programme on task 1 to be able to change the duty cycle while remaining the same frequency (i.e. total time interval which is 1s). It also asks to enable two buttons to change the duty cycle up and down, and those changes should be in steps of 5%. First we introduce back external interrupts which are used in the previous laboratory on SW1 and SW0, and the timer/counter overflow interrupt. Yet, we want to make it possible to change the interval by 5% this time. In the previous task, we introduced CustomCNT and lighted LED until it reached 10. Now we modify programme a bit so that we rename that 10 into dutyCNT, and dutyCNT can be changed by one using swtiches. It can be increased upto 20 and decreased to 1. In other words, each step for both in/decreasing is changing interval by 5% ($1/20 = 0.05$).

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-26
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 4
;     Title: Task 2
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: generate Pulse with modulation(PWM)
;
;     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 initialise

.org OVF0addr
rjmp timer0int
.org INT0addr
rjmp incduty
.org INT1addr
rjmp decduty

.def temp = r16
.def LEDstat = r17
.def CustomCNT = r18
.def dutyCNT = r19
.equ COUNTVALUE = 206
.equ maxCNT = 20

.org 0x72
initialise:
```

```
50
;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
55 out SPL,R20

ldi temp, 0x01 ;Set data direction registers.
out DDRB, temp ;Set B port as output ports
ldi LEDstat, 0x00
60 out portB, LEDstat


;ldi temp, 0x01      ;Set data direction registers.
;out DDRD, temp      ;Set D port as input ports
65
ldi temp, 0b101      ;Setting up a prescaler. See table 42
out TCCR0B, temp
ldi temp, (1<<TOIE0)
sts TIMSK0, temp
70
ldi temp, COUNTVALUE
out TCNT0, temp

ldi temp,0b11
75 out EIMSK, temp
ldi temp, 0b1010
sts EICRA, temp

sei  ;Enable global interrupt
80
ldi CustomCNT,0
ldi dutyCNT, 10

start:    ;The relative jump uses two cycles.
85 rjmp start

timer0int:
push temp
in temp, SREG
90 push temp

;Reset counter value
ldi temp, COUNTVALUE
out TCNT0, temp
95
;;;;
inc CustomCNT
cp dutyCNT,CustomCNT
brlt offLED
100
ldi LEDstat,0x00
rjmp continue

offLED:
105 ldi LEDstat,0xFF

continue:
cpi customCNT, maxCNT
```

```
brne continue2
ldi customCNT,0

continue2: nop

out portB, LEDstat
pop temp
out SREG, temp
pop temp
reti

incduty:
cpi dutyCNT,maxCNT

brge gorikke1
inc dutyCNT

gorikke1: nop
reti

decduty:

cpi dutyCNT,1
brlt gorikke2

dec dutyCNT
gorikke2: nop
reti
```

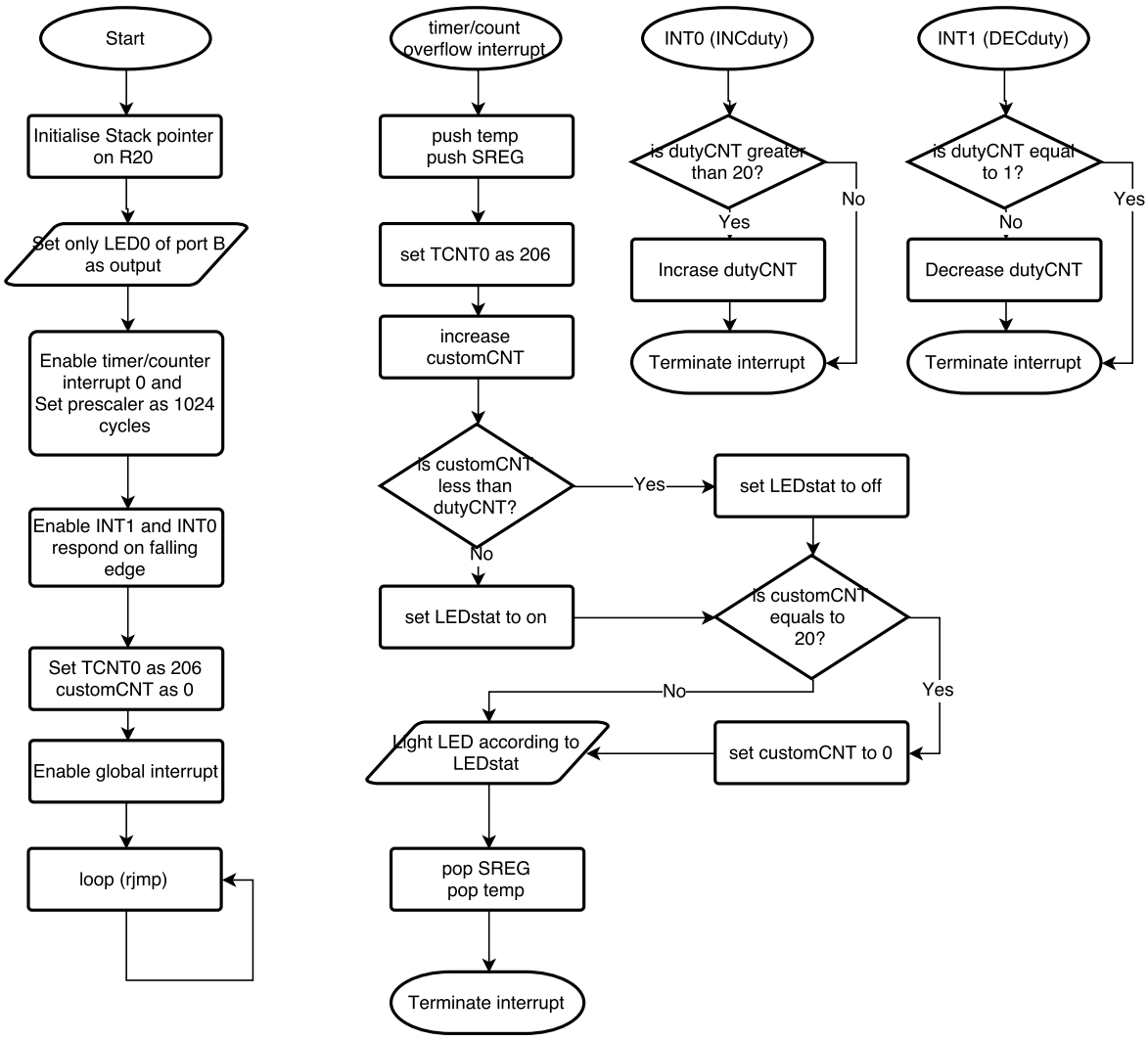

Figure 2: Flowchart of task 2

4 Task 3 and 4

Task 3 and 4 asks to write a programme to use a serial communication port (RS232) using Universal Synchronous and Asynchronous serial Receiver and Transmitter(USART). We have configured the CPU's oscillation frequency to 1.8432MHz as it provides the minimal error rate among the availalbe frequencies. (See Expamples of UBRRn settings table). We have also chosen to set a UBRR baud rate as 2400bps, therefore we are assigning UBRR as 47 in our programme. For this task we are using polled UART, which means that inside our pullchar we are jumping to a label until it receives a signal over the serial port.

We used USART1 (instead of USART0). Thus, we connected RX1 to portD pin2 and TX1 to portD pin3 (it should be on port E in case of using USART0). The task was executed on a terminal programme on Windows called PuTTY, and we would like to notify you that for task 3 we are not having 'putChar' part of the code.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016-09-26
;     Author:
;     Songho Lee
;     Sarpreet Singh
```

```
;
;
;     Lab number: 4
;     Title: Task 4
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Serial communication using polled UART + echo function
;
;     Input ports: none
;     RS232 RXD, TXD are connected to PD2,PD3 respectivly.
;     see alternative functions of port D
;
;     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"
;I am intending to use 1.8432MHz configuration with baud rate as 2400 bps.
;Therefore I set UBRR as 47
.equ UBRR_choice = 47
.def char_in = R16
.def temp = R17

.org 0x00
     rjmp start

.org 0x72
start:

     ;Initialising the output port
     ldi temp, 0xFF ;Set data direction registers.
     out DDRB, temp ;Set B port as output ports

     ;Initialising USART
     ldi temp, UBRR_choice
     sts UBRR1L, temp
     ;Setting enable flags in UCR
     ldi temp, (1<<TXEN1) | (1<<RXEN1)
     sts UCSR1B, temp

pullChar:
     lds Temp, UCSR1A
     sbrs temp, RXC1
     rjmp pullChar
     lds char_in, UDR1

outLED:
     mov temp, char_in
     com temp
     out portB, temp

putChar:
lds temp, UCSR1A
```

```
    sbrs temp, UDRE1
    rjmp putChar

70  sts UDR1,char_in
    rjmp pullChar
```

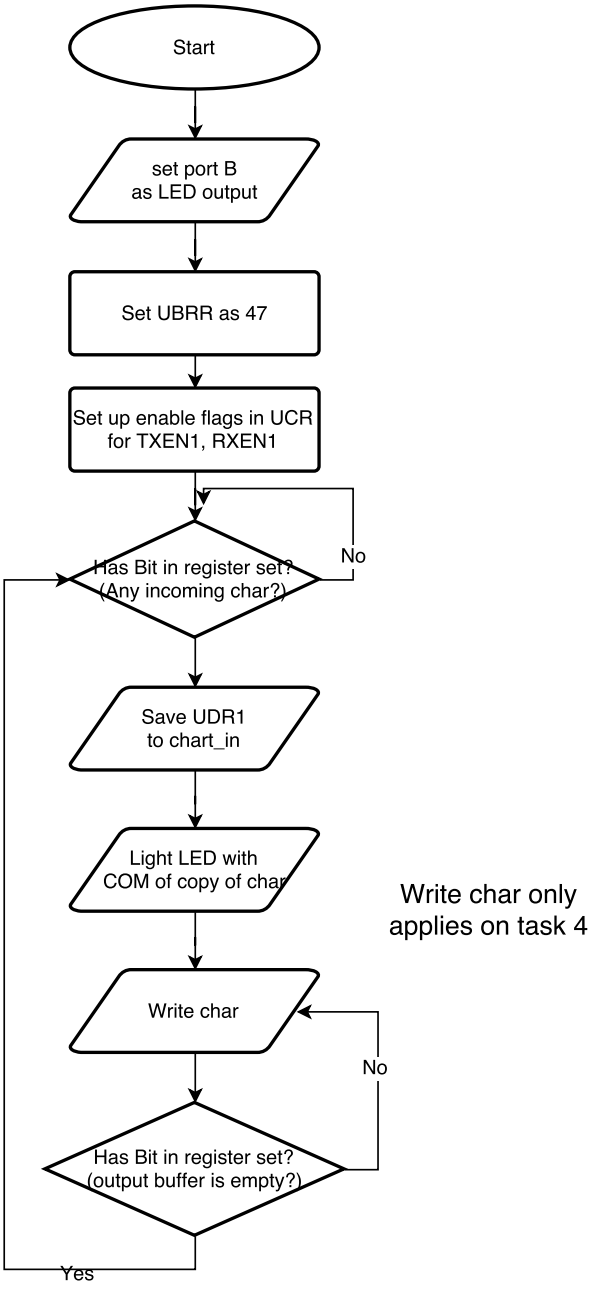

Figure 3: Flowchart of task 3 and 4

5 Task 5

In our task 5 we are modifying the task 3 and 4 into Interrupt based USART. In the previous task, it was polling the input char. It means that it is listening constantly in a loop basis until it receives a signal. From this task, we are using interrupt to respond on the input, instead of constantly checking on the loop. Since we were using USART1, we match our interrupt subroutine to URXC1addr.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;     1DT301, Computer Technology I
;     Date: 2016—09—26
;     Author:
;     Songho Lee
;     Sarpreet Singh
;
;
;     Lab number: 4
;     Title: Task 5
;
;     Hardware: STK600, CPU ATmega2560
;
;     Function: Serial communication using interrupt based UART
;
;     Input ports: none
;     RS232 RXD, TXD are connected to PD2,PD3 respectivly.
;     see alternative functions of port D
;
;     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"
;I am intending to use 1.8432MHz configuration with baud rate as 2400 bps.
;Therefore I set UBRR as 47
.equ UBRR_choice = 47
.def char_in = R16
.def temp = R17

.org 0x00
     rjmp start
.org URXC1addr
     rjmp readChar

.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

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

```
     ;Initialising USART
     ldi temp, UBRR_choice
     sts UBRR1L, temp
     ;Setting enable flags in UCR

     ;ldi temp, 0b10011000
     ldi temp, (1<<TXEN1) | (1<<RXEN1) | (1<<RXCIE1)
     sts UCSR1B, temp
     ; Enable interrupt


     sei  ;Set up global interrupt flag
     loop: nop
     rjmp loop

readChar:
     lds temp, UCSR1A
     lds char_in, UDR1
     rcall outLED
     rcall putChar
     reti

outLED:
     mov temp, char_in
     com temp
     out portB, temp
     ret

putChar:
     lds temp, UCSR1A
     sbrs temp, UDRE1
     rjmp putChar
     sts UDR1,char_in
     ret
```

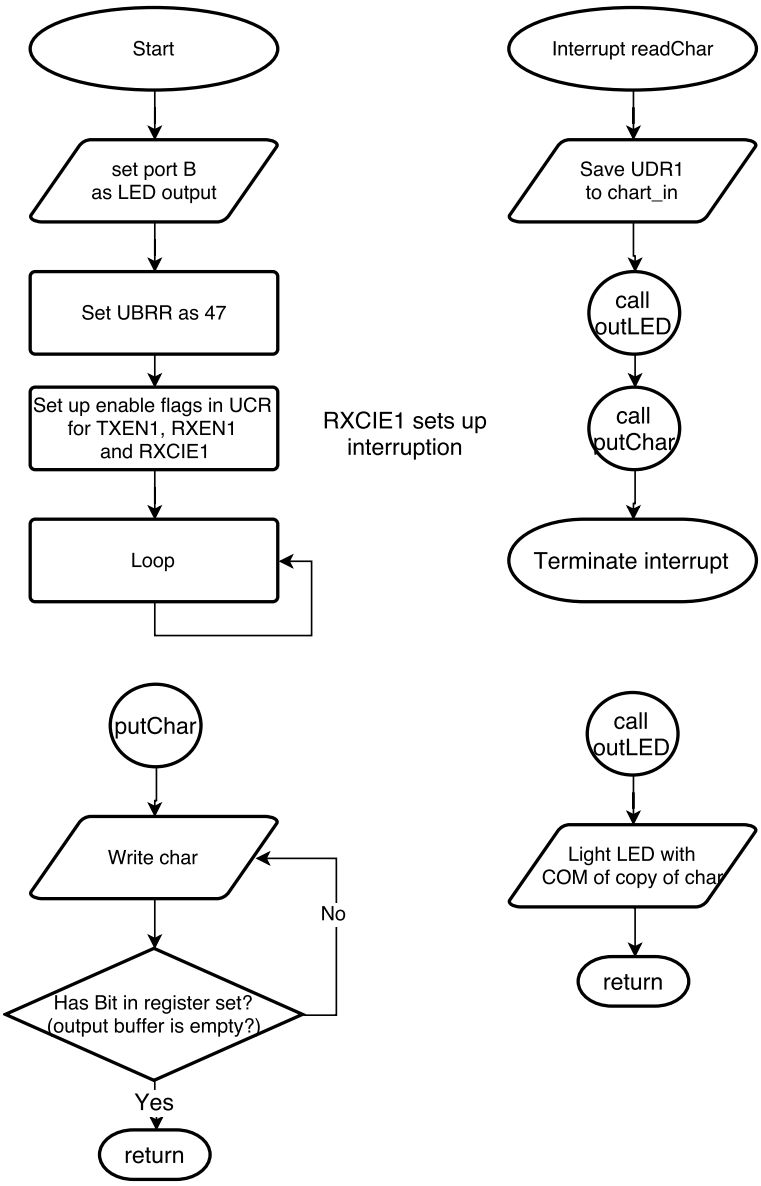

Figure 4: Flowchart of task 5