



## Lab 3 Report



*Authors:* Andrei Neagu  
Konstantinos Tatsis  
*Course code:* 1DT301

**Contents**

<b>1</b>	<b>Task 1</b>	<b>1</b>
1.1	Flowchart 1 . . . . .	2
<b>2</b>	<b>Task 2</b>	<b>3</b>
2.1	Flowchart 2 . . . . .	4
<b>3</b>	<b>Task 3</b>	<b>5</b>
3.1	Flowchart 3 . . . . .	6
<b>4</b>	<b>Task 4</b>	<b>7</b>
4.1	Flowchart 4 . . . . .	8
<b>5</b>	<b>Task 5</b>	<b>9</b>
5.1	Flowchart 5 . . . . .	10

## 1 Task 1

Write a program in Assembly that creates a square wave. One LED should be connected and switch with the frequency 1 Hz. Duty cycle 50. (On: 0.5 sec, Off: 0.5 sec.) Use the timer function to create an interrupt with 2 Hz, which change between On and Off in the interrupt subroutine.

[illegible]

```

out TCNT0, temp
inc counter
cpi counter, 2                ; if counter is 2 then 0,5 sec have passed
breq led                      ; then branch to change_led_state

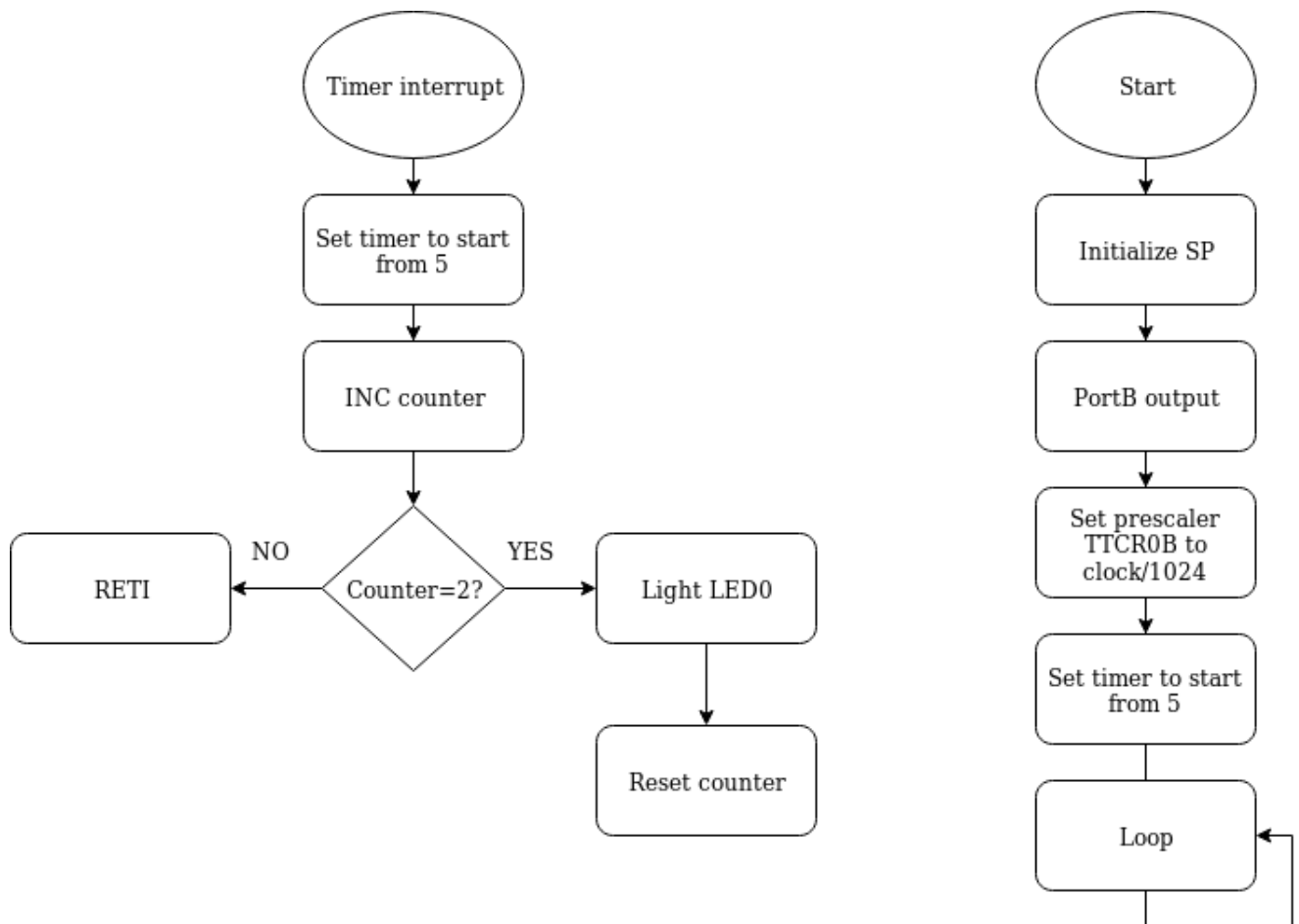
rjmp again

led:
    com ledState              ; toggle LED0
    clr counter              ; reset counter to 0

again:
    pop temp                  ; save sreg in SP
    out SREG, temp
    reti

```

## 1.1 Flowchart 1



Modify the program in Task 1 to obtain Pulse Width Modulation (PWM). The frequency should be fixed, but the duty cycle should be possible to change. Use two push buttons to change the duty cycle up and down. Use interrupt for each pushbutton. The duty cycle should be possible to change from 0 up to 100 in steps of 5. Connect the output to an oscilloscope, to visualize the change in duty cycle

3

```

; set start value for timer so next interrupt occurs after 250 ms
ldi temp, 5
out TCNT0, temp
inc counter
cpi counter, 2                ; if counter is 2 then 0,5 sec have passed
breq led                      ; then branch to change_led_state

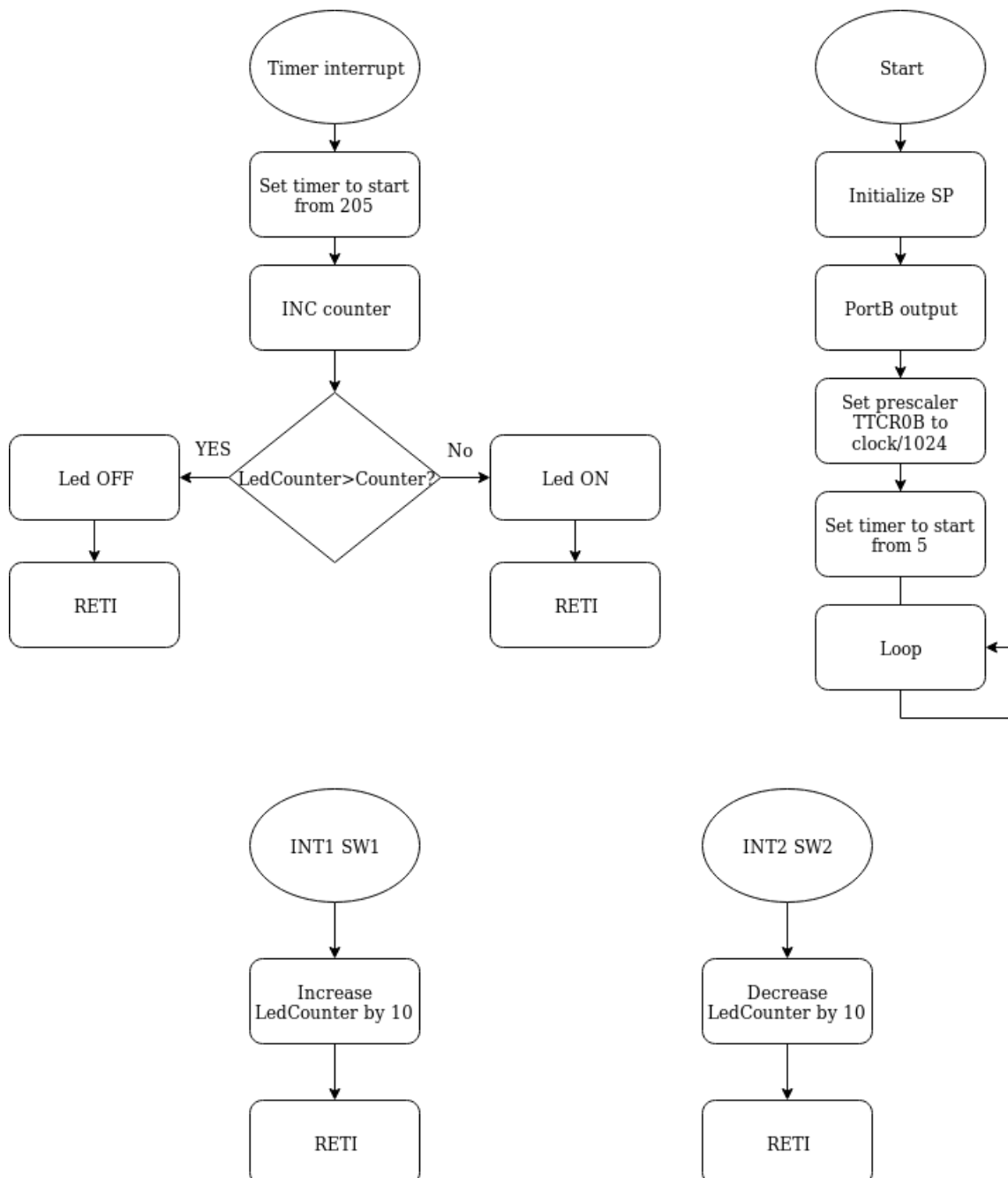
rjmp again

led:
    com ledState              ; toggle LED0
    clr counter              ; reset counter to 0

again:
    pop temp                  ; save sreg in SP
    out SREG, temp
    reti

```

## 2.1 Flowchart 2



### 3 Task 3

Write a program in Assembly that uses the serial communication port0(RS232). Connect a computer to the serial port and use a terminal emulation program. (Ex. Hyper Terminal) The program should receive characters that are sent from the computer, and show the code on the LEDs. For example, if you send character A, it has the hex code 65, the bit pattern is 0110 0101 and should be displayed with LEDs On for each 'one'. Use polled UART, which means that the UART should be checked regularly by the program. Serial communication.

```
;
; Task 3.asm
;
; Created: 2019-10-12 14:35:33
; Author : kt222iq
;
; Replace with your application code
.include "m2560def.inc"

.org 0x00

rjmp start

.org 0x72

start:
ldi r20, 0xFF                ; Load immediately value "0xFF" to r20
out DDRB, r20                ; Set PORTB as output with the value of r20
out PORTB, r20                ; Output value of r20 to PORTB

ldi r20, 12                    ; Store Prescaler value in UBRR1L
sts UBRR1L, r20

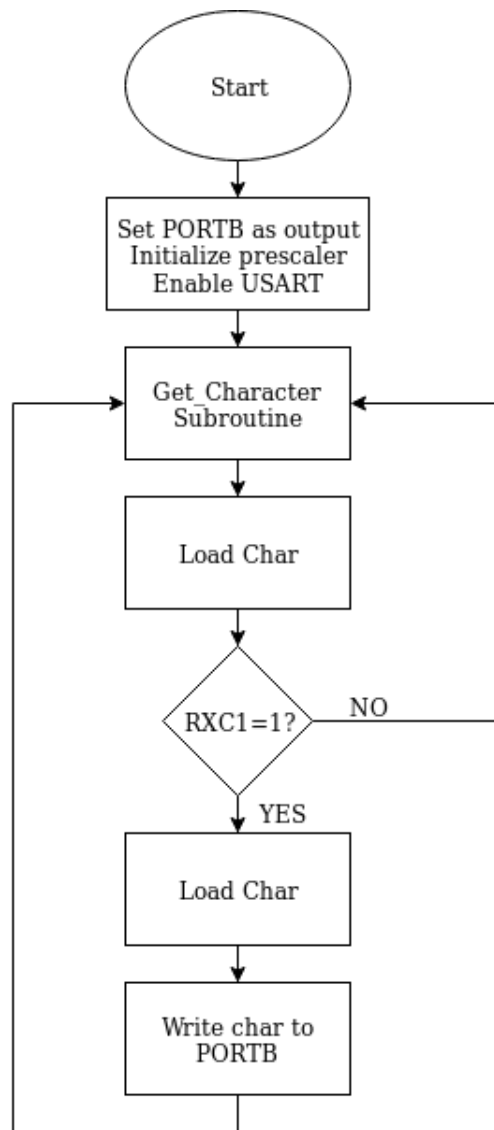
ldi r20, (1<<TXEN1) | (1<<RXEN1) ; Enable receiver and Transmitter
sts UCSRB, r20                ; Set TXEN1 and RXC1 enable flags

main:

get_Character:
lds r20, UCSRA                ; read UCSRA I/O register to r20
sbrc r20, RXC1                ; RXC1 = 1 new Char received
rjmp get_Character            ; RXC1 = 0 then no Char was received
lds r15, UDR1                 ; Read char in UDR1

com r15
out PORTB, r15                ; Write chars to PORTB
com r15
rjmp main
```

### 3.1 Flowchart 3





## 4 Task 4

Modify the program in task 3 to obtain an echo, which means that the received character should also be sent back to the terminal. This could be used as a confirmation in the terminal, to ensure that the character has been transferred correctly.

```
; Task 4.asm
; Created: 2019-10-12 14:38:08
; Author : kt222iq
; Replace with your application code
.include "m2560def.inc"

.org 0x00

rjmp start

.org 0x72

start:

ldi r20, 0xFF                ; Load immediately value "0xFF" to r20
out DDRB, r20                ; Set PORTB as output with the value of r20
out PORTB, r20               ; Output value of r20 to PORTB

ldi r20, 12                  ; Store Prescaler value in UBRR1L
sts UBRR1L, r20

ldi r20, (1<<TXEN1) | (1<<RXEN1) ; Enable receiver and Transmitter
sts UCSRB, r20               ; Set TXEN1 and RXEN1 enable flags

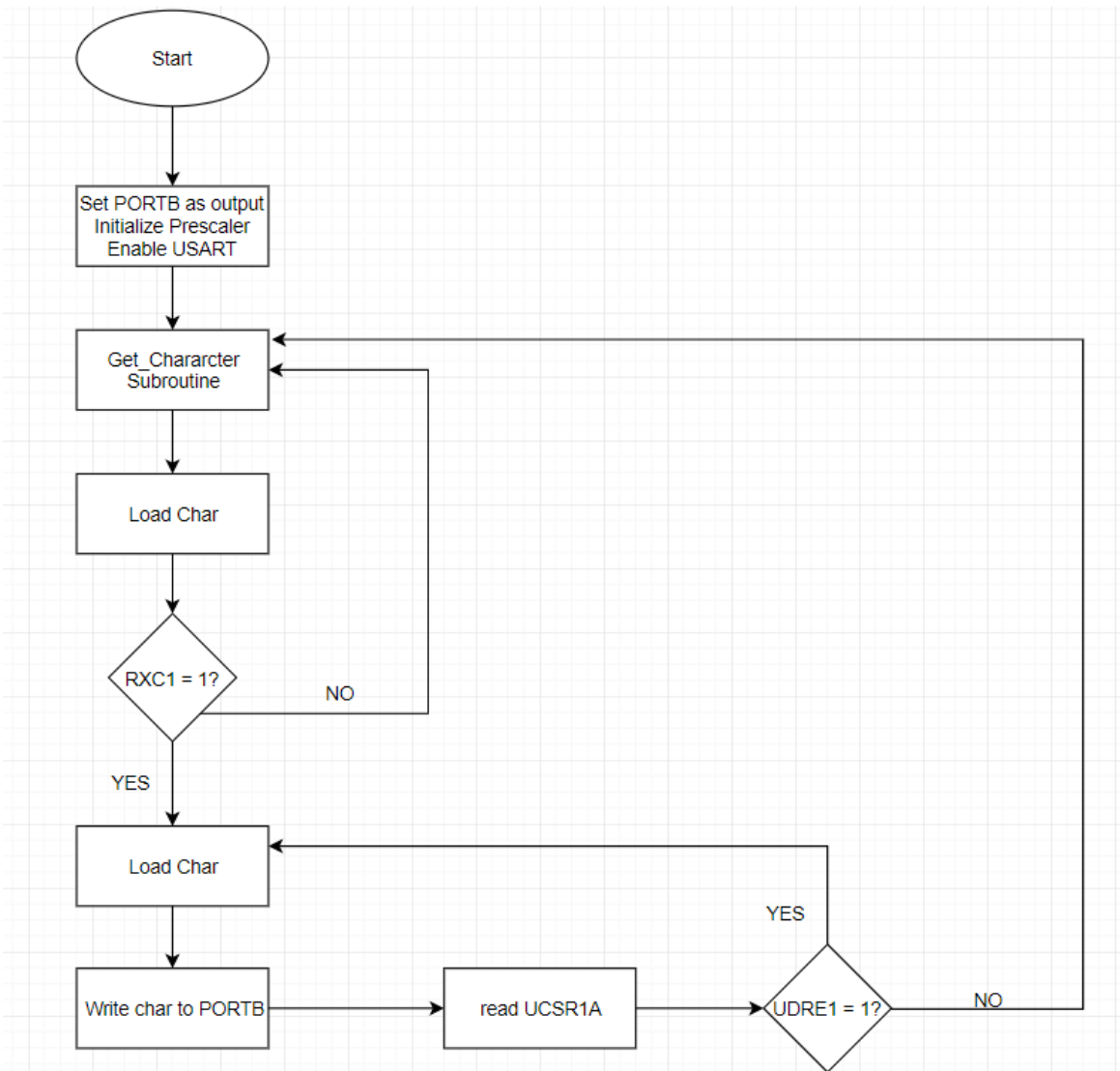
main:

get_Character:
lds r20, UCSRA               ; read UCSRA I/O register to r20
sbrs r20, RXC1               ; RXC1 = 1 new Char received
rjmp get_Character          ; RXC1 = 0 then no Char was received
lds r15, UDR1               ; Read char in UDR1

com r15
out PORTB, r15              ; Write chars to PORTB
com r15

put_Character:
lds r20, UCSRA               ; read UCSRA I/O register to r20
sbrs r20, UDRE1              ; UDRE1 = 1 buffer is empty
rjmp put_Character          ; UDRE1 = 0 buffer is not empty
sts UDR1, r15               ; Write char to UDR1
rjmp get_Character          ; Jump back to loop
```

**4.1 Flowchart 4**



## 5 Task 5

Do task 3 and 4, but use Interrupt instead of polled UART.(USART, Rx Complete, USART Data Register Empty and USART, Tx Complete).

```
.include "m2560def.inc"

.def temp = r16
.def ledState = r17
.def complement = r18
.def dataReceived = r19

.equ TRANSFER_RATE = 12          ;1MHz, 4800 bps
.equ TRUE = 0x01
.equ FALSE = 0x00

.cseg

.org 0x00
rjmp reset

.org URXC1addr
rjmp data_received_interrupt

.org UDRE1addr
rjmp buffer_empty_interrupt

.org 0x72

reset:

ldi temp, LOW(RAMEND)
out SPL, temp
ldi temp, HIGH(RAMEND)
out SPH, temp

ser temp
out DDRB, temp

ldi temp, TRANSFER_RATE
sts UBRR1L, temp

ldi temp, (1<<RXEN1) | (1<<TXEN1) | (1<<RXCIE1) | (1<<UDRIE1)
sts UCSR1B, temp

sei
clr ledState

main_loop:
    rcall led_output
    rjmp main_loop

led_output:
    mov complement, ledState
    com complement
    out PORTB, complement

    ret

data_received_interrupt:
    lds ledState, UDRI             ;load received data to ledState
    ldi dataReceived, TRUE

    reti

buffer_empty_interrupt:
    cpi dataReceived, FALSE
    breq buffer_empty_end

    sts UDRI, ledState            ;send data
    ldi dataReceived, FALSE

buffer_empty_end:
    reti
```

5.1 Flowchart 5

