



Lab 3 Report



Authors: Andrei Neagu
Konstantinos Tatsis
Course code: 1DT301

Contents

1	Task 1	1
1.1	Flowchart 1	2
2	Task 2	3
2.1	Flowchart 2	4
3	Task 3	5
3.1	Flowchart 3	6
4	Task 4	7
4.1	Flowchart 4	8
5	Task 5	9
5.1	Flowchart 5	11

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

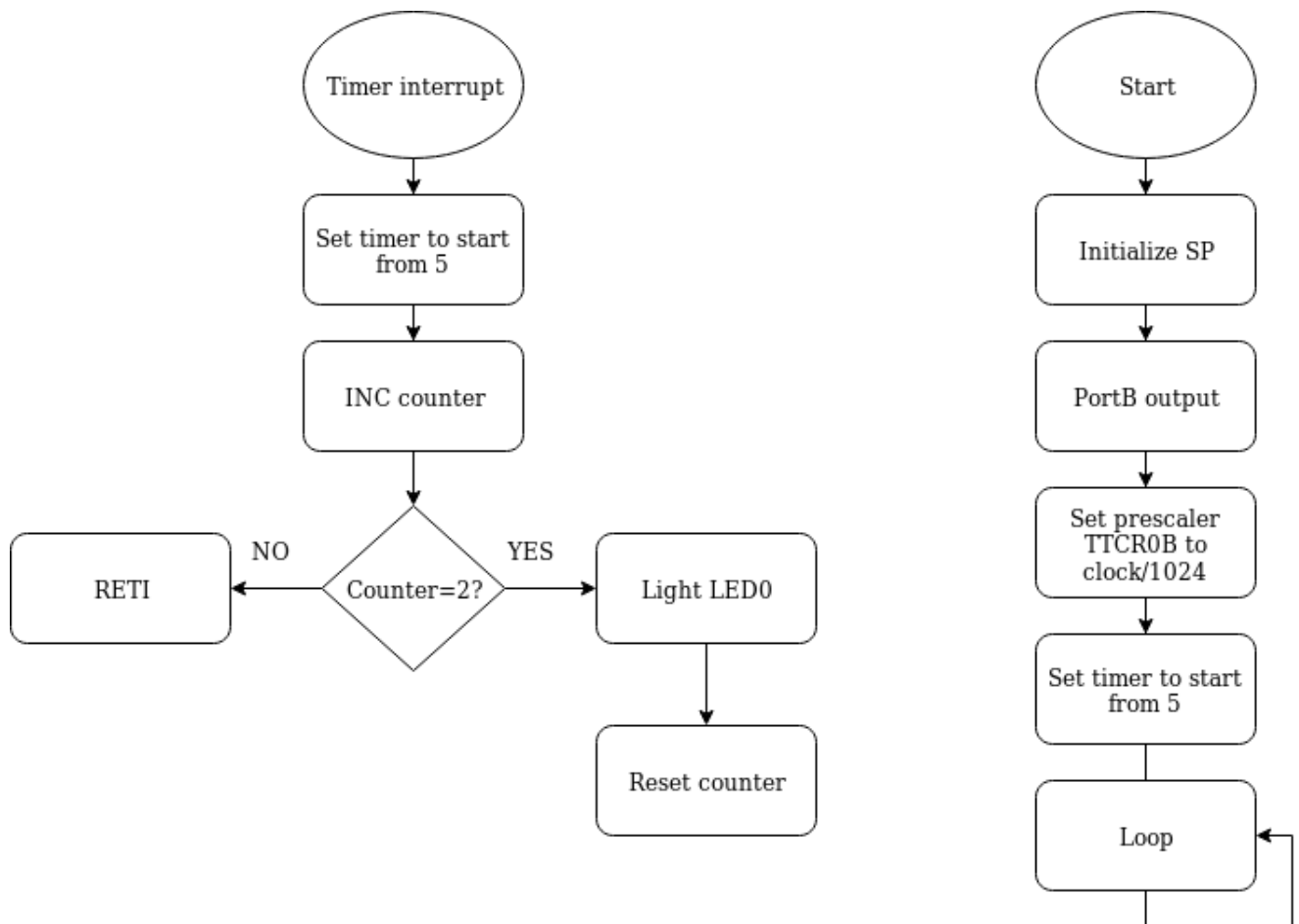
rjmp again

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

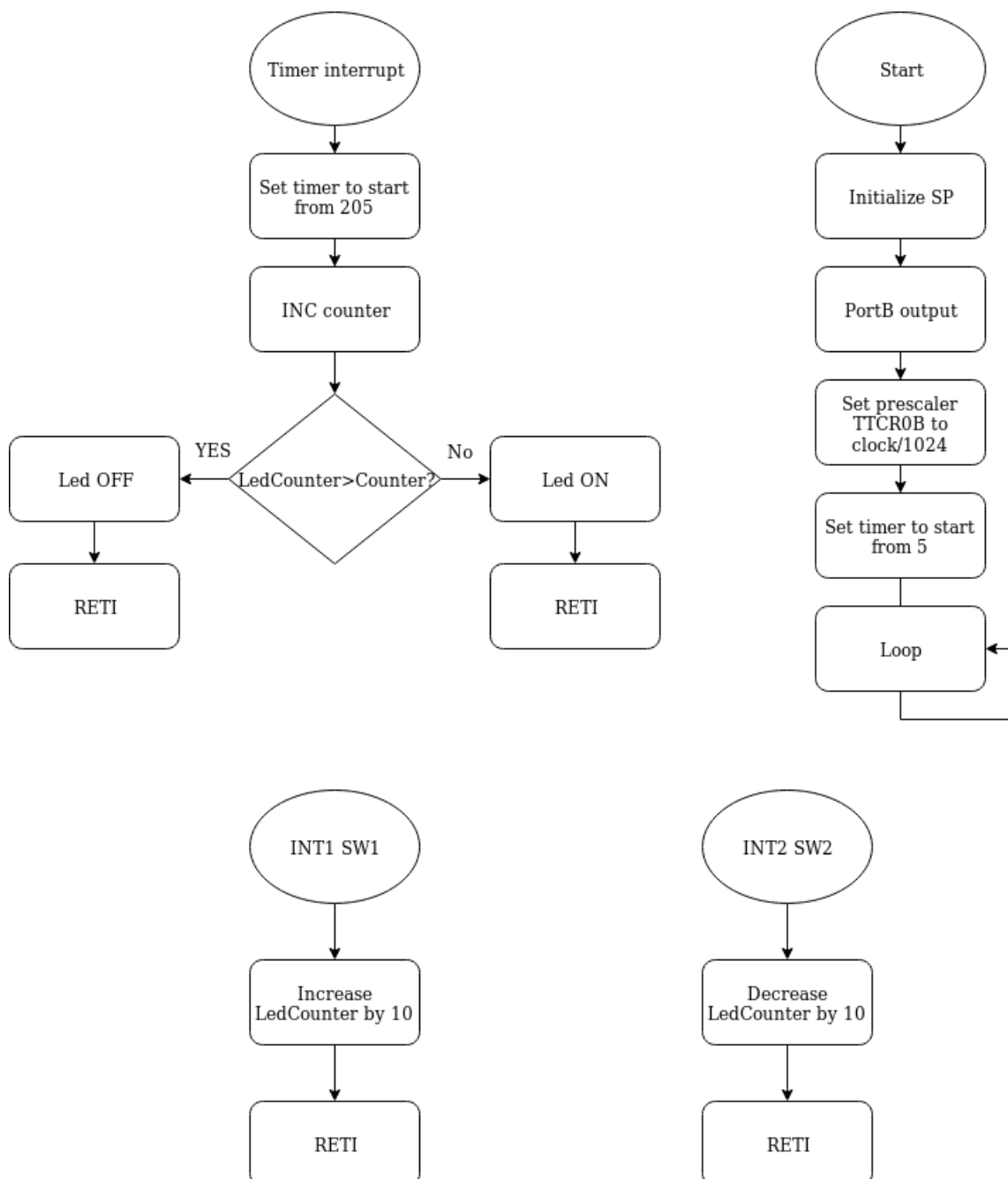
rjmp again

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

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

```

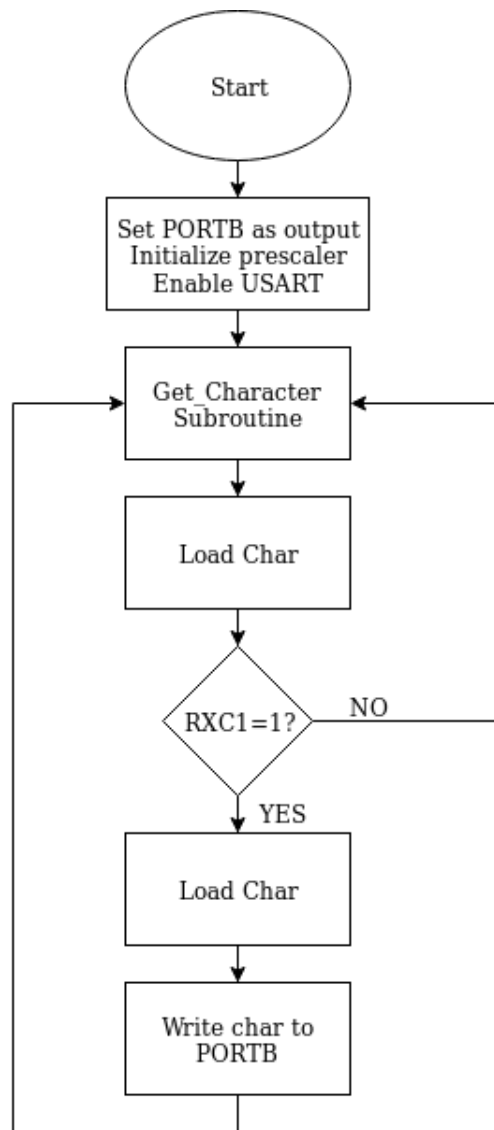
2.1 Flowchart 2



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.

5

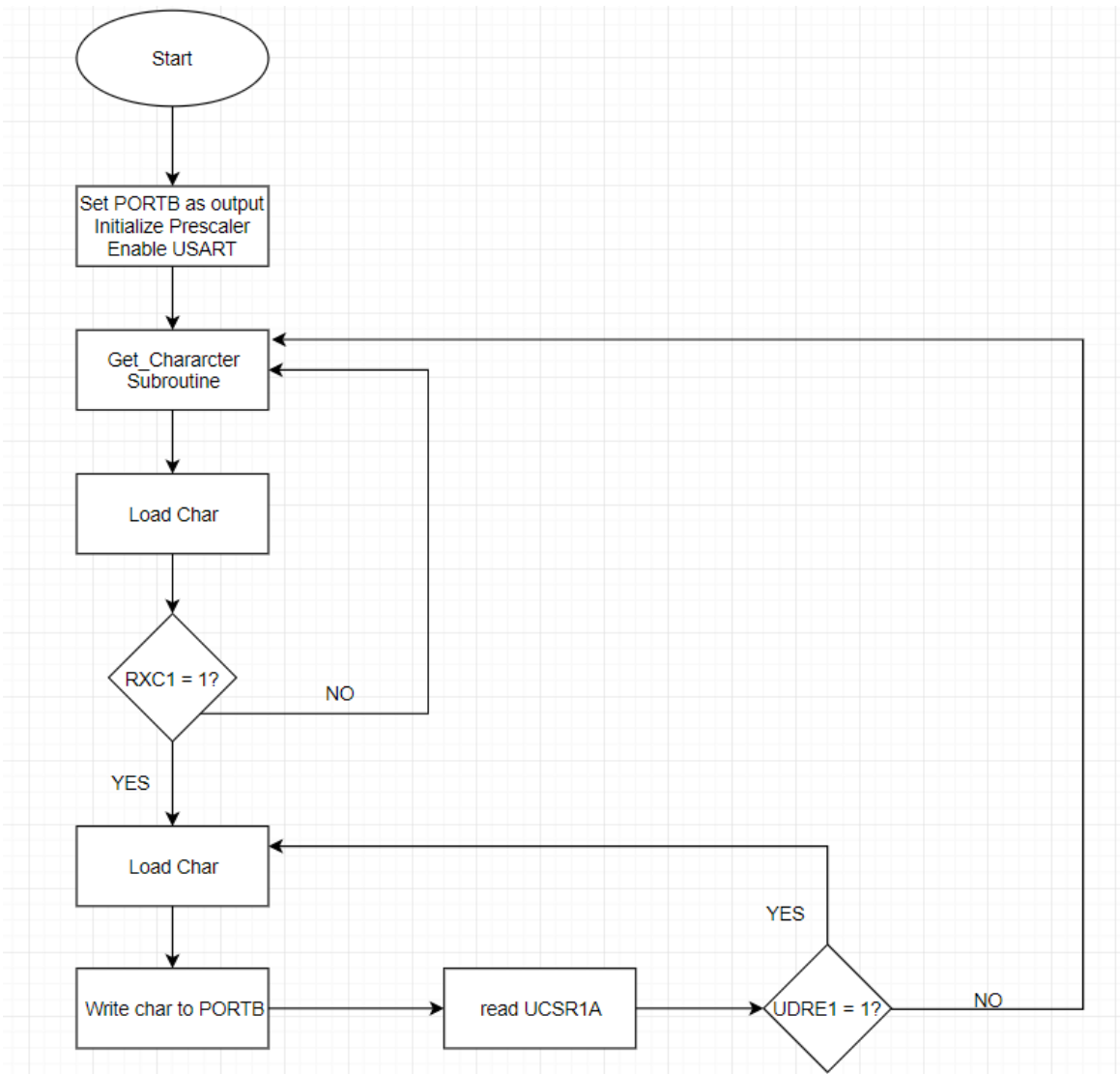
3.1 Flowchart 3



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.

7

4.1 Flowchart 4



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

9

```

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

