

Lab 3 Report



Authors: Andrei Neagu

Konstantinos Tatsis

Course code: 1DT301

Contents

1	Task 1 1.1 Flowchart 1	1 2
2	Task 2 2.1 Flowchart 2	3
3	Task 3 3.1 Flowchart 3	5
4	Task 4 4.1 Flowchart 4	7 8
	Task 5 5.1 Flowchart 5	9 10

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.

```
;>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
 Date: 2019-10-10
; Author:
  Andrei Neagu
 Konstantinos Tatsis
 Lab number: 3
 Title: Timer and UART
; Hardware: STK600, CPU ATmega2560
; Function: Square wave generator
; Input ports:
 Output ports: PORTB
 Subroutines: reset, loop, again, led, timer0_int
; Included files: m2560def.inc
 Other information:
; Changes in program: (Description and date)
.include "m2560def.inc"
.def temp = r16
.def ledState = r17
.def counter = r18
. org 0x00
rjmp reset
. org OVF0addr
rjmp timer0_int
.org 0x72
reset:
ldi temp, LOW(RAMEND) ; initialize stack pointer
out SPL, temp
l\,d\,i\ temp\ ,\ HIGH(RAMEND)
out SPH, temp
ldi temp, 0x01
                              ; set DDRB as output
out DDRB, temp
ldi temp, 0x05
                              ; prescaler value to TCCR0B
out TCCR0B, temp
                              ;0b101 means clock/1025
ldi temp, (1<<TOIE0) ; enable overflow flag
sts TIMSKO, temp
                              ; to TIMSK
ldi temp, 5
                                      ; starts from 5 to 255 meaning 250 times
out TCNT0, temp
                              ; until overflow
sei
clr ledState
                              ; clear ledstate used for outputting to leds
loop:
       out PORTB, ledState
       rjmp loop
timer0\_int:
       in temp, SREG; save sreg in SP
       push temp
       ; set start value for timer so next interrupt occurs after 250 ms
       ldi temp, 5
```

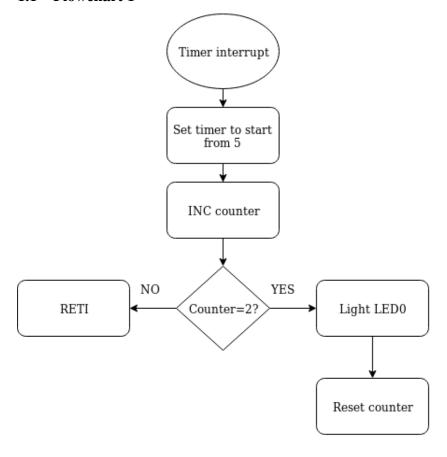
out TCNTO, temp
inc counter
cpi counter, 2 ; if counter is 2 then 0,5 sec have passed
breq led ; then branch to change_led_state

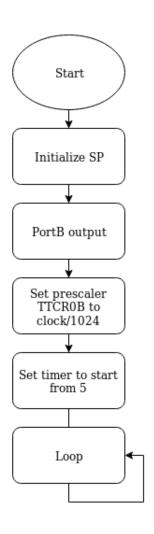
rjmp again
led:
 com ledState ; toggle LEDO
 clr counter ; reset counter to 0

again:

pop temp ; save sreg in SP out SREG, temp

reti

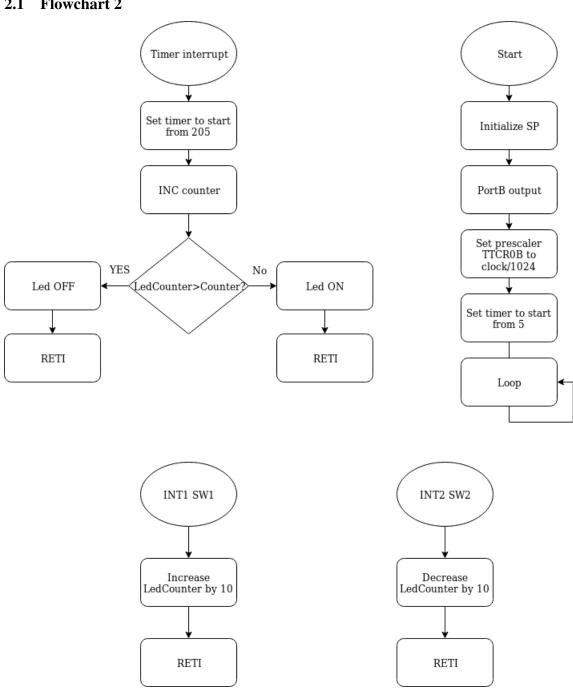




Modify the program in Task 1 to obtain Pulse Width Modulation (PWM). The frequency should be fixed, but the duty cycle should be possible to change. Use two push buttons to change the duty cycle up and down. Use interrupt for each pushbutton. The duty cycle should be possible to change from 0 up to 100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

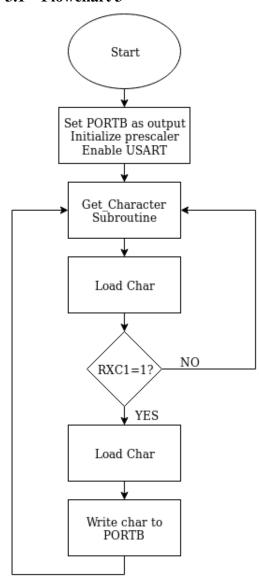
```
;>>>>>>>>>>>>>>>>>>
  1DT301, Computer Technology I
 Date: 2019-10-10
 Author:
  Andrei Neagu
 Konstantinos Tatsis
 Lab number: 3
  Title: Timer and UART
 Hardware: STK600, CPU ATmega2560
; Function: Square wave generator
 Input ports:
 Output ports: PORTB
  Subroutines: reset , loop , again , led , timer0_int Included files: m2560 def.inc
 Other information:
 Changes in program: (Description and date)
.include "m2560def.inc"
.def temp = r16
. def ledState = r17
.def\ counter = r18
.org 0x00
rjmp reset
.org OVF0addr
rjmp timer0_int
. org 0x72
reset:
\begin{array}{ll} 1\,d\,i & temp\;,\;\; LOW(RAMEND)\\ o\,u\,t & SPL\;,\;\; temp \end{array}
                        ; initialize stack pointer
ldi temp, HIGH(RAMEND)
out SPH, temp
ldi temp, 0x01
                                 ; set DDRB as output
out DDRB, temp
ldi temp, 0x05
                                 ; prescaler value to TCCR0B
out TCCR0B, temp
                                 ;0 b101 means clock/1025
                      ; enable overflow flag
ldi temp, (1<<TOIE0)
sts TIMSKO, temp
                                 ; to TIMSK
ldi temp, 5
                                         ; starts from 5 to 255 meaning 250 times
out TCNT0, temp
                                 ; until overflow
clr ledState
                                ; clear ledstate used for outputting to leds
loop:
        out PORTB, ledState
        rjmp loop
timer0_int:
        in temp, SREG ; save sreg in SP
        push temp
```

```
; 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 then led; then branch to change_led_state
rjmp again
led:
         com ledState ; toggle LED0 clr counter ; rese
                                         ; reset counter to 0
again:
         pop temp
out SREG, temp
                                               ; save sreg in SP
         reti
```



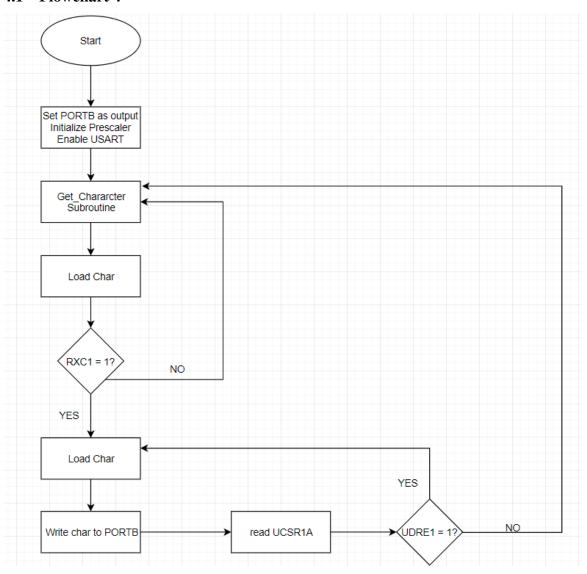
Write a programin Assembly that uses the serial communication port0(RS232).Connect a computer to the serial port and use a terminal emulation program. (Ex. Hyper Terminal)The program shouldreceive characters that are sent from the computer, and show the code on the LEDs. For example, if you send character 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
                                                                        ; Set TXEN1 and RXC1 enable flags
sts UCSR1B, r20
main:
get_Character:
lds r20, UCSR1A
                                   ; read UCSR1A I/O register to r20
sbrs r20, RXC1
rjmp get_Character
lds r15, UDR1
                                   ; RXC1 = 1 new Char received
; RXC1 = 0 then no Char was received
                                    ; Read char in UDR1
com r15
out PORTB, r15
                                    ; Write chars to PORTB
com r15
rjmp main
```



Modify the program in task 3 toobtain 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:
                                    ; Load immediately value "0xFF" to r20
; Set PORTB as output with the value of r20
1\,d\,i\ r20\;,\ 0xFF
out DDRB, r20
out PORTB, r20
                                     ; Output value of r20 to PORTB
1di r20, 12
                                              ; Store Prescaler value in UBRR1L
sts UBRR1L, r20
1di r20, (1<<TXEN1) | (1<<RXEN1)
                                             ; Enable receiver and Transmitter
sts UCSR1B, r20
                                                                ; Set TXEN1 and RXC1 enable flags
main:
get_Character:
lds r20, UCSR1A
                                    ; read UCSR1A I/O register to r20
                                    ; RXC1 = 1 new Char received
sbrs r20, RXC1
rjmp get_Character
lds r15, UDR1
                                    ; RXC1 = 0 then no Char was received
                                     ; Read char in UDR1
out PORTB, r15
                                     ; Write chars to PORTB
com r15
put_Character:
lds r20, UCSR1A
                                    ; read UCSR1A I/O register to r20
                                  ; UDRE1 = 1 buffer is empty
; UDRE1 = 0 buffer is not empty
sbrs r20, UDRE1
rjmp put_Character
sts UDR1, r15
                                    ; Write char to UDR1
rjmp get_Character
                                     ; Jump back to loop
```



Do task 3 and 4, but use Interrupt instead of polledUART.(USART, Rx Complete, USART Data Register Empty andUSART, 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
.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) \mid (1 << TXEN1) \mid (1 << RXCIE1) \mid (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
data\_received\_interrupt:
     lds ledState, UDR1
ldi dataReceived, TRUE
                                               ; load received data to ledState
     reti
buffer_empty_interrupt:
     cpi \ data Received \ , \ FALSE
    breq buffer_empty_end
          sts UDR1, ledState
                                      ; send data
         ldi dataReceived, FALSE
    buffer_empty_end:
         reti
```

