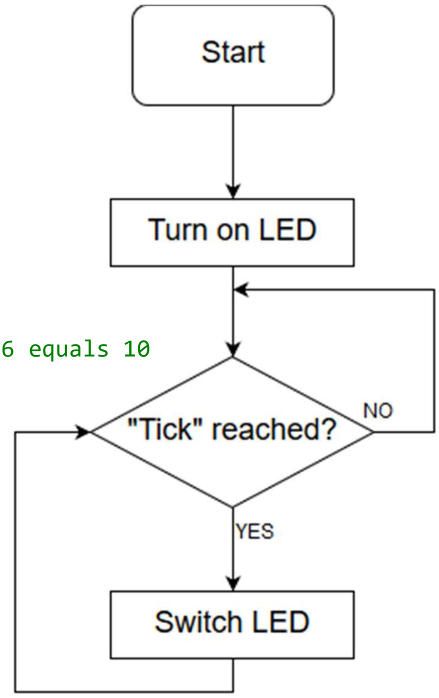
TASK 1

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-10-05
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 4
; Title: Timer and UART.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Square wave generator.
;
; Input ports: None.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information: Clock set at 1MHz.
;
; Changes in program: 2017-10-06.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00
jmp restart
.org OVF0addr               ; address for Timer/Counter0 Overflow interrupt
jmp timer0int

.org 0x72

restart:
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 r16, 0x01               ; set data direction registers.
out DDRB, r16               ; set B port as output ports

ldi r17, 0x00
out PORTB, r17

ldi r16, 0x05               ; setting up prescaler value to TCCR0
out TCCR0B, r16             ; CS2 - CS2 = 101, osc.clock / 1024 -> timer counts every ms.
(1000 times / second)

ldi r16, (1<<TOIE0)         ; timer 0 enable flag, TOIE0
sts TIMSK0, r16             ; to register TIMSK

ldi r16, 206                ; starting value for counter
out TCNT0, r16              ; counter register
```

```
sei                         ; enable global interrupt
ldi r18, 0                  ; help counter

start:
rjmp start                  ; main loop

timer0int:
push r16                    ; timer interrupt routine
in r16, SREG                ; save SREG on stack
push r16
                            ; reset counter value
ldi r16, 206
out TCNT0, r16
inc r18                     ; increment counter
cpi r18, 10                 ; check if "tick" is reached - when r16 equals 10
brne continue
ldi r18, 0
com r17                     ; flip/invert
out PORTB, r17              ; push new state to PORTB

continue:
nop
pop r16                     ; restore SREG
out SREG, r16
pop r16                     ; restore register
reti                        ; return from interrupt

; source: Slides from lecture 7
```

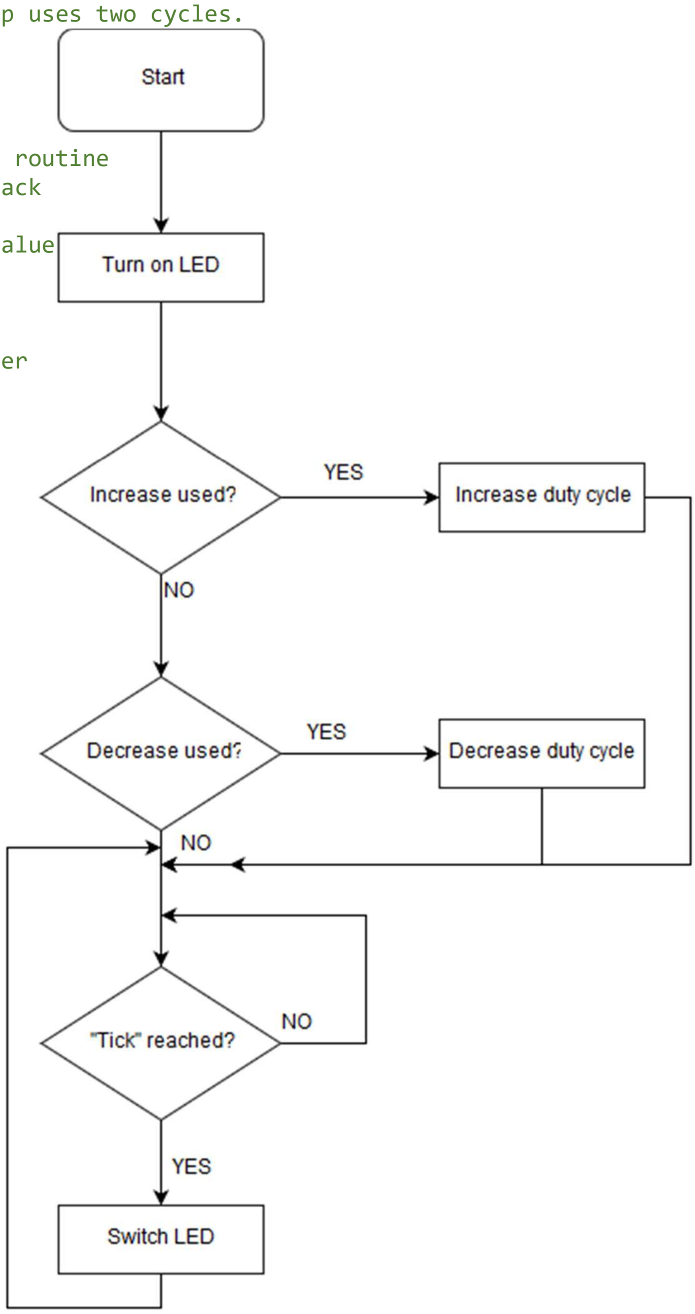


/*Description:
* Here we have a program, which turns a LED on and off with the frequency 1MHz. The Duty cycle is
*50% which means that the LED is 0.5sec on and 0.5sec off. The timer/counter increases on every cycle,
*which is dependent on the Prescaler, and the interrupt is triggered whenever the 8bit counter (TCNT)
*overflows. We needed a second counter, which is increased by one whenever the first counter is
*increased 50x and calls the time interrupt because of the overflow.
*/

Task 2

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-10-05
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 4
; Title: Timer and UART.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Square wave generator.
;
; Input ports: None.
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information: Clock set at 1MHz.
;
; Changes in program: 2017-10-06.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.org 0x00
rjmp restart
.org OVF0addr
rjmp timer0int
.org INT0addr
rjmp increase
.org INT1addr
rjmp decrease

.def LED = r17
.def counter = r18
.def duty_counter = r19
.equ max_counter = 20
.org 0x72

restart:
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 r16, 0x01                      ; set data direction registers.
out DDRB, r16                      ; set B port as output ports

ldi LED, 0x00
out PORTB, LED

ldi r16, 0x04                      ; setting up prescaler value to TCCR0
out TCCR0B, r16                    ; CS2 - CS2 = 101, osc.clock / 1024 -> timer counts
                                   ; every ms (1000 times / second)

ldi r16, (1<<TOIE0)                ; timer 0 enable flag, TOIE0EIMSK 0
sts TIMSK0, r16                    ; to register TIMSK

ldi r16, 205                       ; starting value for counter
out TCNT0, r16                     ; counter register

ldi r16, 0x03                      ; INT0 and INT1 enabled
out EIMSK, r16

ldi r16, 0x0F                      ; falling and rising edge
sts EICRA, r16
sei                                ; enable global interrupt

ldi counter, 0
ldi duty_counter, 10
```

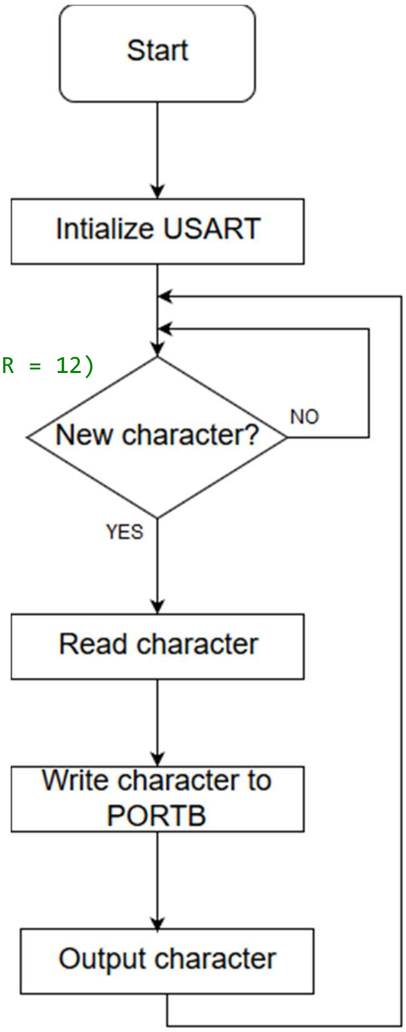
```
start:                            ;The relative jump uses two cycles.
nop
rjmp start

timer0int:
push r16                          ; timer interrupt routine
in r16, SREG                      ; save SREG on stack
push r16
                                  ; reset counter value
ldi r16, 205
out TCNT0, r16

inc counter                       ; increment counter
cp duty_counter, counter
brlt led_off

ldi LED, 0x00
rjmp continue

led_off:
ldi LED, 0xFF

continue:
cpi counter, max_counter
brne continue2
ldi counter, 0

continue2:
nop
out PORTB, LED
pop r16
out SREG, r16
pop r16
reti

increase:
cpi duty_counter, max_counter
brge after_inc
inc duty_counter

after_inc:
nop
reti

decrease:
cpi duty_counter, 1
brlt after_dec
dec duty_counter

after_dec:
nop
reti

; source: Slides from lecture 7
```



/*Description
* This is a modified version of task 1, in which the duty cycle can be increased or decreased by the use of
*an external interrupt for each case. */

TASK 3

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-10-07
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 4
; Title: Timer and UART.
;
; Hardware: STK600, CPU ATmega2560
;
; Function:  Serial communication using polled UART.
;
; Input ports: none.
; Connected RS232 RXD, TXD to PD2, PD3.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information: Clock set at 1MHz.
;
; Changes in program: 2017-10-08.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.equ UBBR_value = 12   ; 4800 as speed (osc.=1MHz, 4800 bps => UBBRR = 12)

.org 0x00
rjmp reset

.org 0x72

reset:
ldi r16, 0xFF           ; PORTB output
out DDRB, r16
ldi r16, 0x55           ; Init val to output
out PORTB, r16

ldi r16, UBBR_value     ; store Prescaler val in UBRR1L
sts UBRR1L, r16         ; connect cable to pin 2/3 on Port D

ldi r16, (1<<TXEN1) | (1<<RXEN1)  ; enable USART transmitter
sts UCSR1B, r16                    ; (set TX and RX enable flags)

main:
get_char:
lds r16, UCSR1A        ; read from USART to get character
sbrs r16, RXC1         ; new character, RXC1=1
```



```
rjmp get_char          ; no char received RXC1=0

lds r17, UDR1          ; read char in UDR
port_output:
com r17                 ; invert bits to show binary on leds
out PORTB, r17          ; write char to PORTB
;com r17

rjmp main

; source: Slides from lecture 7
```

/*Description:

* Here we have a program, which uses a serial communication port (RS232) using Universal

*Synchronous and Asynchronous serial Receiver and Transmitter(USART).

*We use USART1 (instead of USART0). We connected RX1 to PORTD pin2 and TX1 to

*PORTD pin3 (it would be on port E in case of using USART0). We tested this task by executing it on the

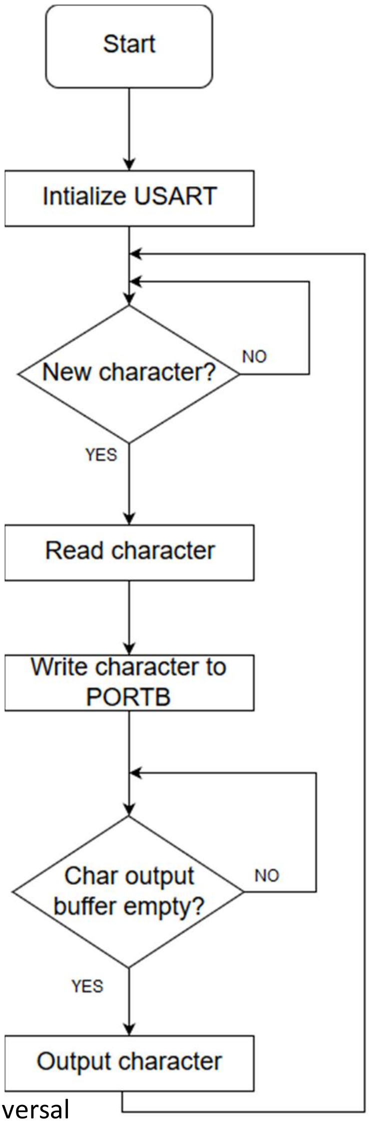
*terminal program called PuTTY.

*/

TASK 4

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-10-07
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 4
; Title: Timer and UART.
;
; Hardware: STK600, CPU ATmega2560
;
; Function:  Serial communication using polled UART.
;
; Input ports: none.
; Connected RS232 RXD, TXD to PD2, PD3.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information: Clock set at 1MHz.
;
; Changes in program: 2017-10-08.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.equ UBBR_value = 12   ; 4800 as speed (osc.=1MHz, 4800 bps => UBBRR = 12)

.org 0x00
rjmp reset
```

```
.org 0x72

reset:
ldi r16, 0xFF               ; PORTB output
out DDRB, r16
ldi r16, 0x55               ; Init val to output
out PORTB, r16

ldi r16, UBBR_value         ; store Prescaler val in UBRR1L
sts UBRR1L, r16                    ; connect cable to pin 2/3 on Port D

ldi r16, (1<<TXEN1) | (1<<RXEN1)  ; enable USART transmitter
sts UCSR1B, r16                    ; (set TX and RX enable flags)

main:
get_char:
lds r16, UCSR1A         ; read from USART to get character
sbrs r16, RXC1          ; new character, RXC1=1
rjmp get_char           ; no char received RXC1=0

lds r17, UDR1           ; read char in UDR

port_output:
com r17                 ; invert bits to show binary on leds
out PORTB, r17          ; write char to PORTB
com r17

put_char:
lds r16, UCSR1A    ;
sbrs r16, UDRE1         ; buffer is empty = UDRE1 = 1
rjmp put_char           ; buffer is not empty = UDRE1 = 0
sts UDR1, char          ; write char to UDR1
rjmp main               ; jump back to loop

; source: Slides from lecture 7
```

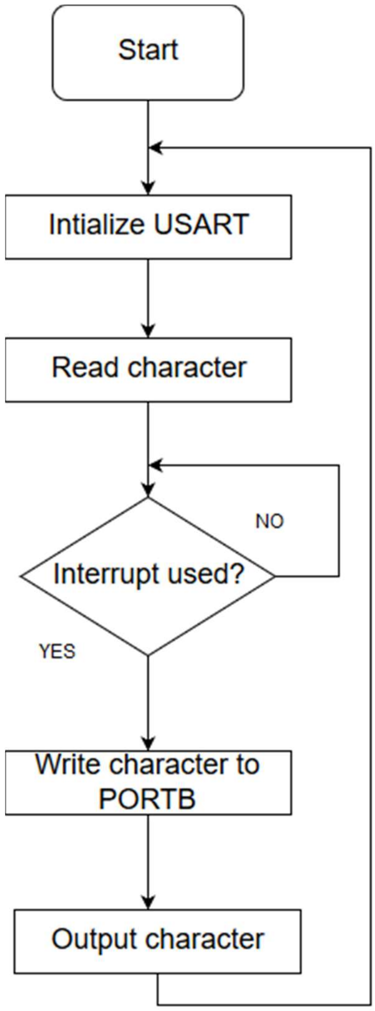


```
/*Description:
* Here we have a program, which uses a serial communication port (RS232) using Universal
*Synchronous and Asynchronous serial Receiver and Transmitter(USART).
*We use USART1 (instead of USART0). We connected RX1 to PORTD pin2 and TX1 to
*PORTD pin3 (it would be on port E in case of using USART0). We tested this task by executing it on the
*terminal program called PuTTY. Furthermore, in this task we are obtaining an echo, which means that
* the received character is also sent back to the terminal.
*/
```

TASK 5

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2017-10-07
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
```

```
;
; Lab number: 4
; Title: Timer and UART.
;
; Hardware: STK600, CPU ATmega2560
;
; Function:  Serial communication using interrupt based UART.
;
; Input ports: none.
; Connected RS232 RXD, TXD to PD2, PD3.
;
; Output ports: On-board LEDs connected to DDRB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information: Clock set at 1MHz.
;
; Changes in program: 2017-10-08.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.equ UBBR_value = 12        ; 4800 as speed (osc.=1MHz, 4800 bps => UBBRR = 12)

.org 0x00
rjmp reset
.org URXC1addr             ; interrupt address
rjmp main

.org 0x72

reset:
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 output port
ldi r16 , 0xFF              ; Set data direction registers
out DDRB , r16              ; PORTB output
ldi r16, 0x55               ; Init val to output
out PORTB, r16

ldi r16 , UBBR_value
sts UBRR1L , r16

;ldi r16, 0b10011000
ldi r16 , (1<< TXEN1 ) | (1<< RXEN1 ) | (1<< RXCIE1 )
sts UCSR1B , r16

sei                         ;Set up global interrupt flag

loop:
nop
rjmp loop

main:
get_char:
```



```
lds r16 , UCSR1A            ; read from USART to get character
lds r17 , UDR1
rcall port_out
rcall put_char
reti                        ; return from interrupt

port_out:
mov r16 , r17
com r16                     ; invert bits to show binary on leds
out PORTB , r16             ; write char to PORTB
ret

/*
com r17
out PORTB, r17
com r17
*/
put_char:
lds r16 , UCSR1A
sbrs r16 , UDRE1            ; buffer is empty = UDRE1 = 1
rjmp put_char               ; buffer is not empty = UDRE1 = 0
sts UDR1 , r17              ; write char to UDR1
ret

; source: Slides from lecture 7
```

```
/*Description:
*  This task is task 3 and 4 modified into an Interrupt based USART. In task 3 and 4 we have been polling
*the input char. In this task we are using Interrupts to answer the input, instead of checking the loop
*constantly.
*/
```