Report

Laboratory 4

October 12, 2017

Author:
Caroline Nilsson *(cn222nd)*
Daniel Alm Grundström *(dg222dw)*
Term: HT 2017
Course: 1DT301 - Computer Technology I

Contents

1 Assignment 1

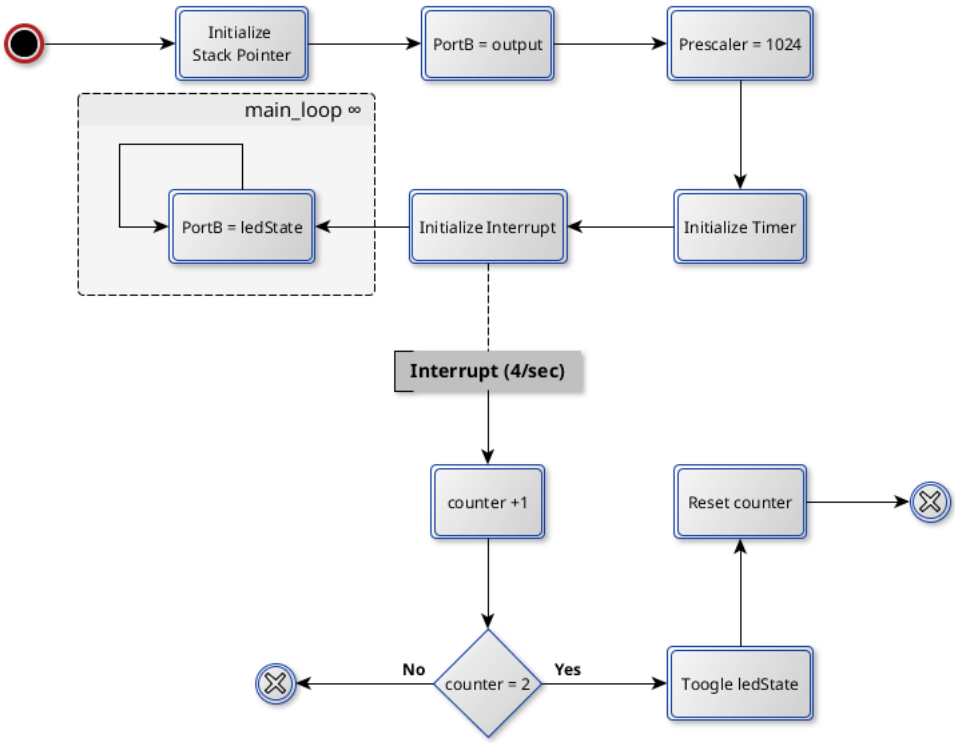

Figure 1: Generate 1 Hz square wave with duty cycle 50%

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-10-08
;    Author:
;                          Caroline Nilsson          (cn222nd)
;                          Daniel Alm Grundström     (dg222dw)
;
;    Lab number:           4
;    Title:                Timer and USART
;
;    Hardware:             STK600, CPU ATmega2560
;
;    Function:             Generates a square wave with a frequency of 1 Hz and a
;                          duty cycle of 50%, which turns LED0 on/off every 1/2
;                          second.
;
;    Input ports:          N/A
;
;    Output ports:         PORTB, PINB0
;
;    Subroutines:          N/A
;    Included files:       m2560def.inc
;
;    Other information:    N/A
;
;    Changes in program:   2017-10-09
;                          Update some comments to make them clearer.
;
;                          2017-10-08
;                          Adds file header with program description.
;
;                          2017-09-25
;                          Implements flowchart design.
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.def temp = r16
.def ledState = r17
.def counter = r18

.equ COMPARISON = 2
.equ PRESCALE = 0x05                ; = 1024, for 1MHz -> 1 count/ms
.equ INIT_TIMER_VALUE = 6           ; counter overflow every 250 ms = 1/4 sec

.CSEG

```

```
48  .ORG 0
49  rjmp reset
50
51  ;Initialize timer overflow interrupt vector
52  .ORG ovf0addr
53  rjmp interrupt
54
55  .ORG 0x72
56
57  reset:
58  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
59  ; Initialize stack pointer
60  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
61  ldi temp, LOW(RAMEND)
62  out SPL, temp
63  ldi temp, HIGH(RAMEND)
64  out SPH, temp
65
66  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
67  ; Initialize PORT B as output
68  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
69  ldi temp, 0x01
70  out DDRB, temp
71
72  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
73  ; Initialize Timer
74  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
75  ;set prescale
76  ldi temp, PRESCALE
77  out TCCR0B, temp
78
79  ;enable overflow flag
80  ldi temp, (1<<TOIE0)
81  sts TIMSK0, temp
82
83  ;set default value for timer
84  ldi temp, INIT_TIMER_VALUE
85  out TCNT0, temp
86
87  sei
88  clr ledState
89
90  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
91  ; Repeatedly outputs ledState while waiting for interrupt
92  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
93  main_loop:
94      out PORTB, ledState
95      rjmp main_loop
96
97  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
98  ; Timer0 overflow interrupt, triggered every 250 ms. Toggles led
99  ; every 2 times the interrupt is triggered.
100 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
101 interrupt:
102     ;save Status Register on stack
103     in temp, SREG
104     push temp
105
106     ;set start value for timer so next interrupt occurs after 250 ms
107     ldi temp, INIT_TIMER_VALUE
108     out TCNT0, temp
109
110     inc counter
111
112     cpi counter, COMPARISON          ; if counter = 2 (1/2 seconds has passed)
113     breq change_led_state            ;     then branch to change_led_state
114
115     rjmp end
116
117     change_led_state:
118         com ledState                 ; toggle LED0
119         clr counter                  ; reset counter to 0
120
121     end:
122         pop temp
123         out SREG, temp
124         reti
```

2 Assignment 2

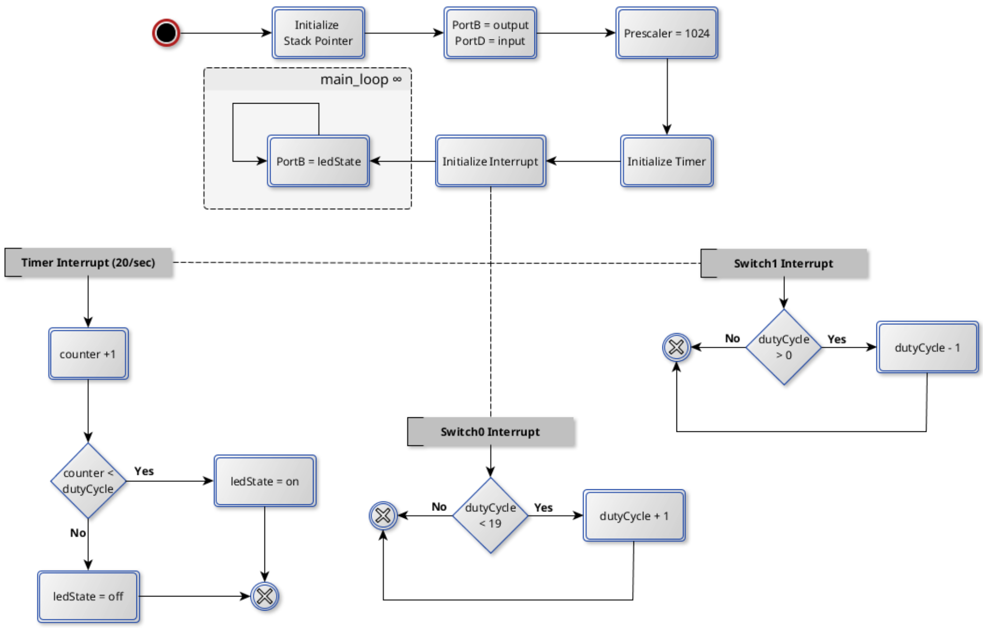

Figure 2: Generate 1 Hz square wave with variable duty cycle 0%-100%

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2017-10-09
;   Author:
;                       Caroline Nilsson          (cn222nd)
;                       Daniel Alm Grundström     (dg222dw)
;
;   Lab number:         4
;   Title:              Timer and USART
;
;   Hardware:           STK600, CPU ATmega2560
;
;   Function:           Builds on assignment 1 to implement a Pulse Width
;                       Modulator (PWM) which allows the duty cycle of the
;                       wave to be changed by pressing SW0 or SW1.
;
;                       The duty cycle starts at 50% (On: 0.5 sec, Off: 0.5 sec)
;                       and can be increased in steps of 5% by pressing SW0 and
;                       decreased in steps of 5% by pressing SW1.
;
;   Input ports:        PORTC, PINC0 & PINC1
;
;   Output ports:       PORTB, PINB0
;
;   Subroutines:        led_out - outputs complement of ledState to PORT B
;   Included files:     m2560def.inc
;
;   Other information:  N/A
;
;   Changes in program: 2017-10-09
;                       Adds file header with program description.
;
;                       2017-09-26
;                       Implements flowchart design.
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.def temp = r16
.def ledState = r17         ; Value to be written to LEDs
.def complement = r18       ; Used when writing to LEDs
.def counter = r19          ; Keeps track of phase of cycle

; Current duty cycle. Can have a value between 0 (0%)
; and 19 (100%), each increment adds 5% to the duty cycle
.def dutyCycle = r20

.equ INIT_TIMER_VALUE = 206             ; 256 - 50 = 206
.equ LED_OFF = 0x00
.equ LED_ON = 0x01
.equ DUTY_CYCLE_MIN = 0
.equ DUTY_CYCLE_MAX = 20

.CSEG

```

```
55  ; Initialize starting point for program
56  .ORG 0
57  rjmp reset
58
59  ; Initialize timer overflow interrupt vector
60  .ORG OVF0ADDR
61  rjmp timer_interrupt
62
63  ; Initialize SW0 interrupt vector
64  .ORG INT0ADDR
65  rjmp sw0_interrupt
66
67  ; Initialize SW1 interrupt vector
68  .ORG INT1ADDR
69  rjmp sw1_interrupt
70
71  .ORG 0x72
72
73  reset:
74  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
75  ;Initialize Stack Pointer
76  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
77  ldi temp, LOW(RAMEND)
78  out SPL, temp
79  ldi temp, HIGH(RAMEND)
80  out SPH, temp
81
82  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
83  ; Initialize ports
84  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
85  ;PortB, Pin0 = output
86  ldi temp, 0x01
87  out DDRB, temp
88
89  ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
90  ; Initialize timer
91  ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
92  ; Set prescale to 1024
93  ldi temp, 0x05
94  out TCCR0B, temp
95
96  ; Enable interrupt on timer overflow
97  ldi temp, (1<<TOIE0)
98  sts TIMSK0, temp
99
100 ; Set default value for timer
101 ldi temp, INIT_TIMER_VALUE
102 out TCNT0, temp
103
104 ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
105 ; Initialize switch interrupts
106 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
107 ; Enable interrupts for SW0 and SW1
108 ldi temp, (3 << INT0)
109 out EIMSK, temp
110
111 ; Trigger interrupts on falling edge (switch released)
112 ldi temp, (3 << ISC00) | (3 << ISC10)
113 sts EICRA, temp
114
115 sei
116 clr ledState
117 ldi dutyCycle, 9
118
119 ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
120 ; main_loop
121 ;       Repeatedly write to LEDs while waiting for
122 ;       interrupts.
123 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
124 main_loop:
125     rcall led_out
126     rjmp main_loop
127
128 ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
129 ; led_out
130 ;       Output complement of current LED state.
131 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
132 led_out:
133     mov complement, ledState
134     com complement
135     out PORTB, complement
136     ret
137
138 ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
139 ; timer_interrupt
140 ;       Interrupt called when a timer overflow occurs,
141 ;       which is set up to occur every 50 ms.
142 ;
143 ;       Compares the value of a counter to the current
144 ;       duty cycle. Depending on if the counter value is
145 ;       higher or lower, the LED state will turn off or on.
146 ;
147 ;       For example, if the duty cycle is set to 4 (20%),
148 ;       then the LED state will be ON 4 out of the 20
149 ;       iterations this interrupt is triggered per second,
150 ;       which means that the LED will be on for 0.2
151 ;       seconds and off for 0.8 seconds.
152 ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
153 timer_interrupt:
154     ; Save Status Register on stack
155     in temp, SREG
```

```
156         push temp
157
158         ; Reset starting value for timer
159         ldi temp, INIT_TIMER_VALUE
160         out TCNT0, temp
161
162         cpi counter, DUTY_CYCLE_MAX
163         brlo compare_duty_cycle
164
165         ldi counter, DUTY_CYCLE_MIN
166
167         compare_duty_cycle:
168         cp counter, dutyCycle       ; if counter < dutyCycle
169         brlo set_led_on             ;     then turn LED on
170         rjmp set_led_off            ; else turn LED off
171
172         set_led_on:
173             ldi ledState, LED_ON
174             rjmp timer_int_end
175
176         set_led_off:
177             ldi ledState, LED_OFF
178
179         timer_int_end:
180             inc counter
181             pop temp
182             out SREG, temp
183             reti
184
185     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
186     ; sw0_interrupt
187     ;       Triggered when switch 0 is pressed Increments the
188     ;       duty cycle if it's not already at DUTY_CYCLE_MAX
189     ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
190     sw0_interrupt:
191         lds temp, PORTD
192
193         sw0_loop:
194         ldi  r31, 130
195         ldi  r30, 222
196     L1: dec  r30
197         brne L1
198         dec  r31
199         brne L1
200         nop
201
202         lds r29, PORTD
203         cp temp, r29
204         brne sw0_loop
205
206         cpi dutyCycle, DUTY_CYCLE_MAX ; If dutyCycle == 20
207         breq sw0_int_end              ;     then skip to end
208
209         inc dutyCycle                 ; else increment dutyCycle
210
211         sw0_int_end:
212             ldi temp, 0x00
213             sts EIFR, temp
214             reti
215
216     ;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
217     ; sw1_interrupt
218     ;       Triggered when switch 1 is pressed. Decrements the
219     ;       duty cycle if it's not already at DUTY_CYCLE_MIN
220     ;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
221     sw1_interrupt:
222         lds temp, PORTD
223
224         sw1_loop:
225         ldi  r31, 130
226         ldi  r30, 222
227     L2: dec  r30
228         brne L2
229         dec  r31
230         brne L2
231         nop
232
233         lds r29, PORTD
234         cp temp, r29
235         brne sw1_loop
236
237         cpi dutyCycle, DUTY_CYCLE_MIN ; If dutyCycle == 0
238         breq sw0_int_end              ;     then skip to end
239
240         dec dutyCycle                 ; else decrement dutyCycle
241
242         sw1_int_end:
243             ldi temp, 0x00
244             sts EIFR, temp
245             reti
```

3 Assignment 3

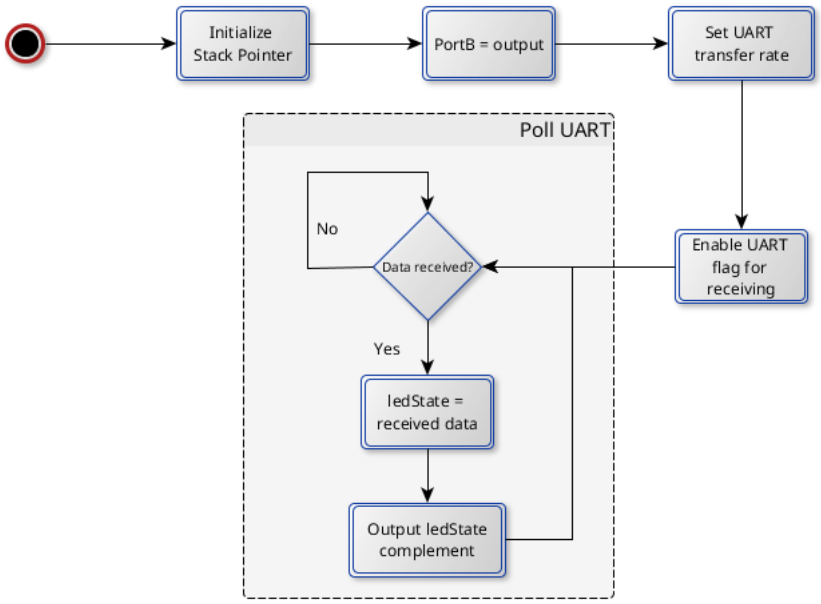

Figure 3: Poll UART and output binary ascii

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-10-12
;    Author:
;                         Caroline Nilsson          (cn222nd)
;                         Daniel Alm Grundström     (dg222dw)
;
;    Lab number:          4
;    Title:               Timer and USART
;
;    Hardware:            STK600, CPU ATmega2560
;
;    Function:            Program that polls the serial port for input characters
;                         and outputs them in ASCII binary to PORTB.
;
;    Input ports:         RS232
;
;    Output ports:        PORTB
;
;    Subroutines:         led_output - outputs complement of register ledState
;                                      to PORTB
;    Included files:      m2560def.inc
;
;    Other information:   Putty is used to enter characters on the computer.
;
;    Changes in program:  2017-10-12
;                         Adds comment header.
;
;                         2017-09-27
;                         Implements flowchart design.
;
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

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

.equ TRANSFER_RATE = 12      ;1MHz, 4800 bps


;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;Initialize Stack Pointer
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, LOW(RAMEND)
out SPL, temp
ldi temp, HIGH(RAMEND)
out SPH, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize port
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ser temp
```

```
out DDRB, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize Serial Communication
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, TRANSFER_RATE
sts UBRR1L, temp            ;set transfer rate

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Enable recieve data
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, (1<<RXEN1)
sts UCSR1B, temp            ;enable UART flag for receiving

clr ledState
rcall led_output

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Check for received data
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
main_loop:
    lds temp, UCSR1A
    sbrs temp, RXC1         ;if RXC flag is clear
        rjmp main_loop      ;then jump to start

    lds ledState, UDR1      ;load received data to ledState

    flush_finished:
    rcall led_output
    rjmp main_loop

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Output ledState to PortB
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
led_output:
    mov complement, ledState
    com complement
    out PORTB, complement

    ret
```

4 Assignment 4

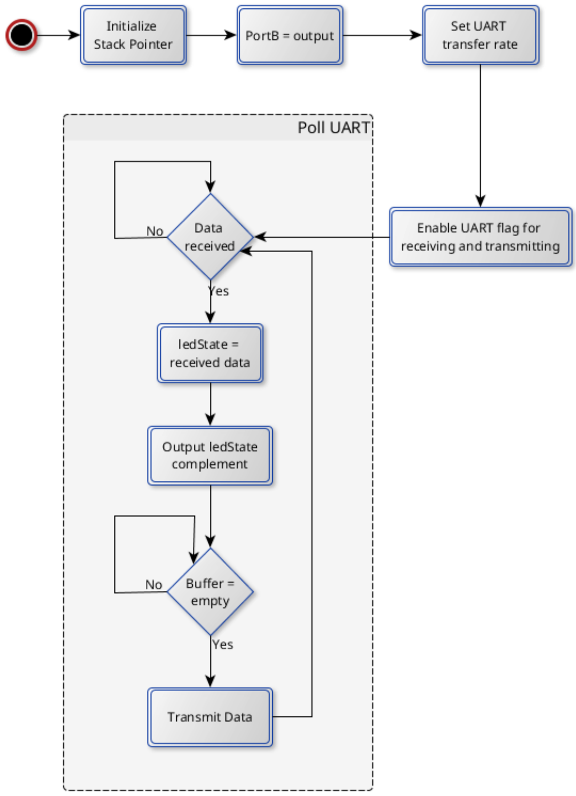

Figure 4: Poll UART, output binary ascii and echo

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-10-12
;    Author:
;                         Caroline Nilsson          (cn222nd)
;                         Daniel Alm Grundström     (dg222dw)
;
;    Lab number:          4
;    Title:               Timer and USART
;
;    Hardware:            STK600, CPU ATmega2560
;
;    Function:            Program that polls the serial port for input characters
;                         and outputs them in ASCII binary to PORTB. In addition,
;                         the received character is echoed back to the receiver.
;
;    Input ports:         RS232
;
;    Output ports:        PORTB, RS232
;
;    Subroutines:         led_output - outputs complement of register ledState
;                                      to PORTB
;    Included files:      m2560def.inc
;
;    Other information:   Putty is used to enter characters on the computer.
;
;    Changes in program:  2017-10-12
;                         Adds comment header.
;
;                         2017-09-27
;                         Implements flowchart design.
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

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

.equ TRANSFER_RATE = 12      ;1MHz, 4800 bps

.org 0
rjmp reset

.org 0x72
reset:
```

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;Initialize Stack Pointer
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, LOW(RAMEND)
out SPL, temp
ldi temp, HIGH(RAMEND)
out SPH, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize port
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ser temp
out DDRB, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize Serial Communication
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, TRANSFER_RATE
sts UBRR1L, temp            ;set transfer rate

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Enable recieve data and transfer data
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, (1<<TXEN1) | (1<<RXEN1)
sts UCSR1B, temp            ;enable UART flag for receiving
                            ;and transmitting

clr ledState
rcall led_output
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Check for received data
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
main_loop:
    lds temp, UCSR1A
    sbrs temp, RXC1         ;if RXC flag is clear
        rjmp main_loop      ;then jump to start

    lds ledState, UDR1      ;load received data to ledState

    rcall led_output

    ldi  r31, 6
    ldi  r30, 19
    ldi  r29, 174
L1: dec  r29
    brne L1
    dec  r30
    brne L1
    dec  r31
    brne L1
    rjmp PC+1


    echo:
        lds temp, UCSR1A
        sbrs temp, UDRE1    ;if transferbuffer !set
            rjmp echo       ;then jump to echo

        sts UDR1, ledState  ;send data

    rjmp main_loop

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Output ledState to PortB
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
led_output:
    mov complement, ledState
    com complement
    out PORTB, complement

    ret
```

5 Assignment 5

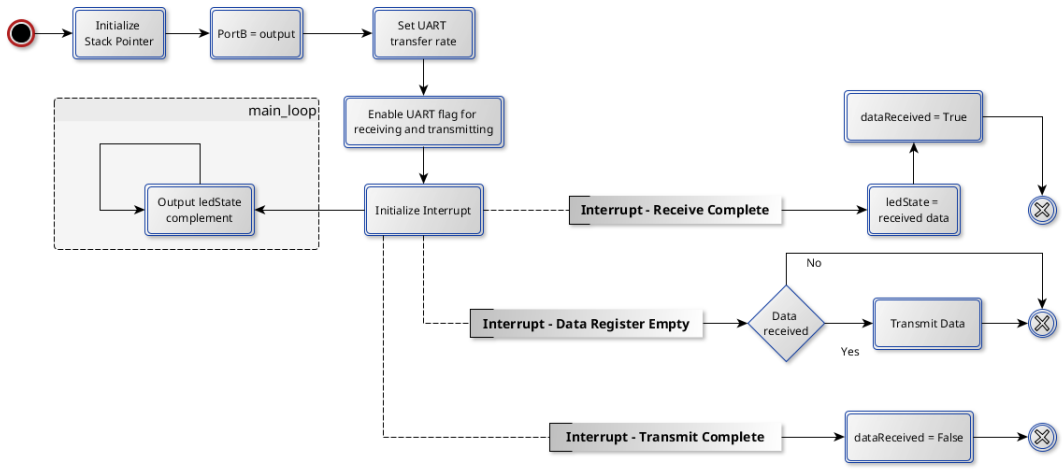

Figure 5: Read UART, output binary ascii and echo. Using interrupts.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2017-10-12
;   Author:
;                       Caroline Nilsson            (cn222nd)
;                       Daniel Alm Grundström       (dg222dw)
;
;   Lab number:         4
;   Title:              Timer and USART
;
;   Hardware:           STK600, CPU ATmega2560
;
;   Function:           Same as assignment 4, but uses interrupts.
;
;   Input ports:        RS232
;
;   Output ports:       PORTB, RS232
;
;   Subroutines:        led_output - outputs complement of register ledState
;                                    to PORTB
;   Included files:     m2560def.inc
;
;   Other information:  Putty is used to enter characters on the computer.
;
;   Changes in program: 2017-10-12
;                       Adds comment header.
;
;                       2017-09-27
;                       Implements flowchart design.
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

.def temp = r16
.def ledState = r17
.def complement = r18
.def dataReceived = r19      ; flag set when data has been
                             ; received

.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 - called on program start and on reset interrupts
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
reset:
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;Initialize Stack Pointer
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, LOW(RAMEND)
out SPL, temp
```

```
ldi temp, HIGH(RAMEND)
out SPH, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize port
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ser temp
out DDRB, temp

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; Initialize Serial Communication
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
ldi temp, TRANSFER_RATE
sts UBRR1L, temp            ;set transfer rate

ldi temp, (1<<RXEN1) | (1<<TXEN1) | (1<<RXCIE1) | (1<<UDRIE1)
sts UCSR1B, temp            ;enable UART flag for receiving
                            ;and transmitting

sei
clr ledState

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; main_loop
;       Outputs ledState while waiting for interrupts
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
main_loop:
    rcall led_output
    rjmp main_loop

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; led_output
;       Output complement of ledState to PortB
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
led_output:
    mov complement, ledState
    com complement
    out PORTB, complement

    ret

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; data_received_interrupt
;       Triggered when data has been received through
;       UART 0. Loads the received data into ledState
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
data_received_interrupt:
    lds ledState, UDR1      ;load received data to ledState
    ldi dataReceived, TRUE

    reti

;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; buffer_empty_interrupt
;       Triggered when UART data register is empty. Checks
;       if dataReceived flag is set and in that case echoes
;       it back to the receiver.
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
buffer_empty_interrupt:
    cpi dataReceived, FALSE
    breq buffer_empty_end

    sts UDR1, ledState  ;send data
    ldi dataReceived, FALSE

    buffer_empty_end:
        reti
```