;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 4, task 1

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; Timer Interrupt: generate a Square wave with 1Hz. 50% Duty cycle.

; Used subroutine:

; Global subroutines (that can be used from other programs):

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.INCLUDE"m2560def.inc"

.CSEG

.org 0x00

jmp start

.org OVF0addr

jmp timer0\_int

start: ;ST-POINTER

ldi r20, HIGH(RAMEND)

out SPH,R20

ldi R20, low(RAMEND)

out SPL,R20

ldi r25, 0xf0 ; initialize DDRB

out DDRB, r25

ldi r20, 0x05 ; prescaler value to TCCR0B

out TCCR0B, r20 ; CS2 - CS2 = 101, osc.clock / 1024

ldi r20,(1<<TOIE0) ; Timer 0 enable flag, TOIE0

sts TIMSK0, r20 ; to register TIMSK

ldi r20,246 ; starting value for counter

out TCNT0, r20 ; counter register

sei ; enable global interrupt

ldi r29, 0xf0

ldi r23, 0x00

wait:

jmp wait

timer0\_int:

inc r23

cpi r23,0b00110010 ; COMPARE IT TO 50

breq on

brne nothing

on:

com r29

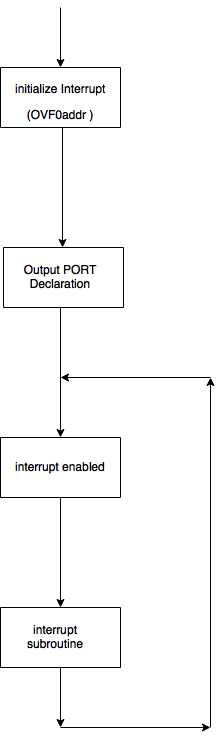
out PORTB, r29 ; AFTER 50 TIMES IT WILL GOES TO OUTPUT

ldi r23, 0x00

nothing:

ldi r20,246 ; starting value for counter

out TCNT0, r20 ; counter register



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 4, task 2

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; PWM: fixed frequency! Duty cycle is dependent of two interrupts one

; decrements duty cycle and the other increments it.

; Used subroutine:

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.org 0x00

jmp restart

.org 0x02

rjmp interrupt\_0 ; Interrupt\_0

.org 0x04

rjmp interrupt\_1 ; Interrupt\_1

.org OVF0addr ; timer\_interuupt

rjmp timer0\_int

.org 0x72

restart:

;ST-POINTER

ldi r16, low(ramend)

out SPL, r16

ldi r16, high(ramend)

out sph, r16

ldi r16 , 0

out DDRD, r16 ; INPUT

ldi r16, 0xff

out ddrb, r16 ; OUTPUT

out ddre, r16

out PortB, r16

.def temp1 = r16

.def temp2 = r19

ldi temp1, 0x05 ; prescaler value to TCCR0

out TCCR0B, temp1 ; CS2 - CS2 = 101, osc.clock / 1024

ldi temp1, (1<<TOIE0) ; Timer 0 enable flag, TOIE0

sts TIMSK0, temp1 ; to register TIMSK

.def on = r20 ; The registers on and off, frequency for the output

; frequency is 100%, the difference between r20 and 255 is 20 and

; difference between r21 and 255 is 0

.def off = r21 ; frequency is 0 %, the difference between r20 and 255 is 0

;and difference between r21 and 255 is 20

.def current\_lights = r22 ; 0 for on and 1 for off

ldi on, 10

ldi off, 10

ldi current\_lights, 0

ldi temp1,245 ; counter = 10

out TCNT0, temp1 ; counter register

ldi r17, 0 ;LEDs on and off in every 50 timer\_interrupt

ldi temp1, 0b00000011 ; Setting Interrupt

out EIMSK, temp1

ldi temp1, 0b00001111

sts EICRA, temp1

ldi r18, 0 ; Output

out PORTB, r18

sei ; enable global interrupt

start:

rjmp start ; main loop

timer0\_int:

push temp1 ; timer interrupt routine

in temp1, SREG ; save SREG on stack

push temp1

inc r17 ; inc r17

cpi r17, 50

brne return ; switch the LEDs after 50 times around timer0\_interrupt

cpi current\_lights, 0

breq turn\_lights\_off

cpi current\_lights, 1

breq turn\_lights\_on

turn\_lights\_off:

ldi temp2, 0xff

out PortB, temp2 ; Turn the LEDs off

out Porte, temp2

ldi current\_lights, 1 ; checking if lEDs are on

ldi r17, 0 ; Reset times counter

rjmp return

turn\_lights\_on:

ldi temp2, 0

out PortB, temp2

out Porte, temp2

ldi current\_lights, 0 ; checking if lEDs are off

ldi r17, 0 ; Reset times counter

return:

rcall set\_TCNT0

pop temp1 ; restore SREG

out SREG, temp1

pop temp1 ; restore register

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

set\_TCNT0: ; timer counter according to the duty cycle

cpi current\_lights, 0

breq zero

cpi current\_lights, 1

breq one

zero:

ldi temp2, 255

sub temp2, off

out TCNT0, temp2

rjmp return\_from\_subroutine

one:

ldi temp2, 255

sub temp2, on

out TCNT0, temp2

return\_from\_subroutine:

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

interrupt\_0: ; Timer interrupt increment duty cycle

push temp1

in temp1, SREG ; Save SREG on stack

push temp1

cpi off, 20

breq keep\_LEDs\_on ; duty cycle == 100%, LEDs on. Timer0\_interrupt disabled

cpi on, 20

breq enable\_timer0\_interuupt0 ; LEDs off, enable Timer0\_interrupt

back\_0:

inc off

dec on

return\_0:

pop temp1 ; restore SREG

out SREG, temp1

pop temp1 ; restore register

reti

keep\_LEDs\_on:

ldi temp2, 0

out PortB, temp2

sts TIMSK0, temp2

rjmp return\_0

enable\_timer0\_interuupt0:

ldi temp2, (1<<TOIE0) ; Timer 0 enable flag, TOIE0

sts TIMSK0, temp2 ; to register TIMSK

rjmp back\_0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

interrupt\_1:

push temp1 ; timer interrupt routine

in temp1, SREG ; save SREG on stack

push temp1

cpi on, 20

breq keep\_LEDs\_off ; If the duty cycle is at 0%, keep all LEDs off and disable Timer0\_interrupt

cpi off, 20

breq enable\_timer0\_interuupt1 ; If all the LEDs are on, re-enable Timer0\_interrupt

back\_1:

inc on

dec off

return\_1:

pop temp1

out SREG, temp1

pop temp1

reti

keep\_LEDs\_off:

ldi temp2, 0xff

out PortB, temp2

ldi temp2, 0

sts TIMSK0, temp2

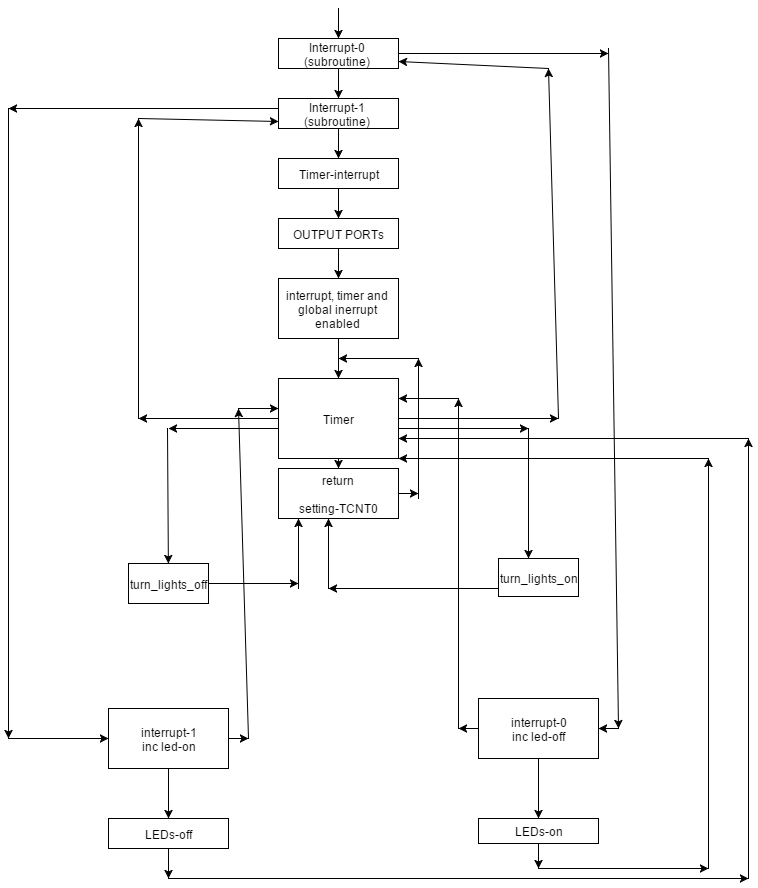
rjmp return\_0

enable\_timer0\_interuupt1:

ldi temp2, (1<<TOIE0) ; enable flag, TOIE0

sts TIMSK0, temp2 ; to register TIMSK

rjmp back\_1



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 4, task 3

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; Is about serial communication port0 (RS232). The computer is connected to the serial port

; and use a terminal emulation program. (Ex. Hyper Terminal). It sends byte (characters) to the

; hyber terminal.

; Used subroutine:

; Global subroutines (that can be used from other programs):

; Other information:

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.def Temp = r17

.def char = r16

.equ UBRRvalue = 12 ; osc.= 1Mhz, 4800 bps => UBBRR = 12. SET ON also 1Mhz on THE HARDWEAR

.org 0x00

rjmp start

.org 0x30

start:

ldi Temp, 0xFF ; PORTB outputs

out DDRB, Temp

ldi Temp, 0x55 ; Initial value to outputs

out PORTB, Temp

ldi Temp, UBRRvalue ; store pescaler value in UBRRIL

sts UBRR1L, Temp ; USART Baud Rate Registers

ldi Temp, (1<<TXEN1) | (1<<RXEN1)

sts UCSR1B, Temp ; set TX and RX enable flags

Getchar:

; recive data

lds Temp, UCSR1A ; read UCSR1A I/O register to r20

sbrs Temp, RXC1 ; RXC1 = 1 => new character

rjmp GetChar ; RXC1 = 0 => no character received

lds Char, UDR1 ; read character in UDR

Port\_output:

com Char

out PORTB, Char ; write character to PORTB

com Char

PutChar: ; send data

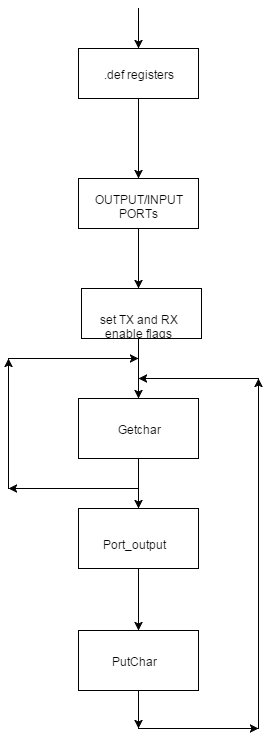
lds Temp, UCSR1A ; read UCSR1A I/O register to r20

sbrs Temp, UDRE1 ; UDRE1 = 1 => buffer is empy

rjmp PutChar ; UDRE1= 0 => buffer is not UDR1

sts UDR1, Char ; write character to UDR1

rjmp GetChar ; return to loop



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 4, task 4

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; Serial communication with echo. It konforms that the characters are send corectly

; Used subroutine:

; Global subroutines (that can be used from other programs):

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.def Temp = r17

.def char = r16

.equ UBRRvalue = 12 ; osc.= 1Mhz, 4800 bps => UBBRR = 12. SET ON also 1Mhz on THE HARDWEAR

.org 0x00

rjmp start

.org 0x30

start:

ldi Temp, 0xFF ; PORTB outputs

out DDRB, Temp

ldi Temp, 0x55 ; Initial value to outputs

out PORTB, Temp

ldi Temp, UBRRvalue ; store pescaler value in UBRRIL

sts UBRR1L, Temp ; USART Baud Rate Registers

ldi Temp, (1<<TXEN1) | (1<<RXEN1)

sts UCSR1B, Temp ; set TX and RX enable flags

Getchar:

; recive data

lds Temp, UCSR1A ; read UCSR1A I/O register to r20

sbrs Temp, RXC1 ; RXC1 = 1 => new character

rjmp GetChar ; RXC1 = 0 => no character received

lds Char, UDR1 ; read character in UDR

Port\_output:

com Char

out PORTB, Char ; write character to PORTB

com Char

PutChar: ; send data

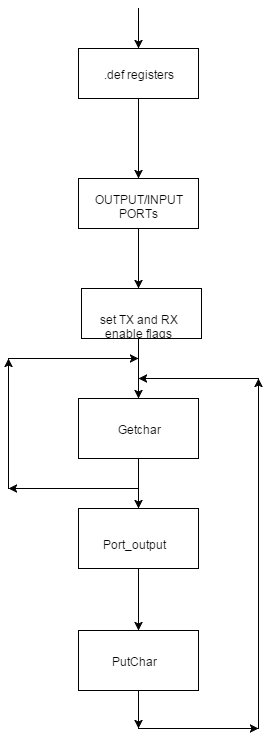
lds Temp, UCSR1A ; read UCSR1A I/O register to r20

sbrs Temp, UDRE1 ; UDRE1 = 1 => buffer is empy

rjmp PutChar ; UDRE1= 0 => buffer is not UDR1

sts UDR1, Char ; write character to UDR1

rjmp GetChar ; return to loop



;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

; ED2022, Computer Technology I

; Lab 4, task 5

; Hardware: STK600, CPU ATmega2560

; Date: 2015-11-26

; Authors: Ivan Hussein

; Function:

; --------

; Serial communication using Interrupt instead of polled UART (RX-TX).

; From task 3 and 4.

; Used subroutine:

; Global subroutines (that can be used from other programs): Included

; Other information: Stock Pointer

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"

.def Temp = r17

.def char = r16

.equ speed = 12 ; osc.= 1Mhz, 4800 bps => UBBRR = 12

.org 0x00

rjmp start

.org URXC1addr

rjmp get\_char

.org 0x50

start:

ldi Temp, LOW(RAMEND)

out SPL, Temp

ldi Temp, HIGH(RAMEND)

OUT SPH, Temp

ldi Temp, 0xFF ; PORTB outputs

out DDRB, Temp

ldi Temp, 0x55 ; Initial value to outputs

out PORTB, Temp

ldi Temp, speed ; store pescaler value in UBRRIL

sts UBRR1L, Temp ; USART Baud Rate Registers

ldi Temp, 0b10011000

;ldi Temp, (1<<TXEN1) | (1<<RXEN1)

sts UCSR1B, Temp ; set TX and RX enable flags

sei

nop\_loop: ; Main loop

nop

rjmp nop\_loop

get\_char:

; recive data

lds r20, UCSR1A ; read UCSR1A I/O register to r20

lds Char, UDR1 ; read character in UDR

output:

com Char

out PORTB, Char ; write character to PORTB

com Char

Put\_Char: ; Echo back to terminal program

lds r20, UCSR1A ; Read UCSR1A I/O register to r20

sts UDR1, Char ; Write character to UDR1

reti

