Lab 4 Summary

Schuster, Mark

Section 1540

10/21/2017

**B)** Pre-Lab Answers:

1. **Pins 3 & 4 of port D.**
2. **2Mbps** , Any larger and BSEL becomes negative.
3. In serial communications, data is transmitted one bit at a time whereas in parallel communications data is transmitted multiple over multiple transmission lines at the same time.
4. Synchronous serial communication mean that the data is transmitted according to a clock, whereas Asynchronous serial communication utilizes extra utility bits (start, stop) to communicate when a packet is being transmitted.
   1. USARTD0\_DATA – Holds the transmission and received data
   2. USARTD0\_STATUS – Holds the status bits, namely RX, TX, & DR flags.
   3. USARTD0\_CTRLA – Used to enable and set the priority levels of the interrupt levels for the port’s USART.
   4. USARTD0\_CTRLB – Used to enable the TX & RX pins, and enable different USART modes.
   5. USARTD0\_CTRLC - Used to set the mode, size, parity, and number of stop bits.
   6. USARTD0\_BAUDCTRLA & B – Used to set the USART’s baud rate by setting the BSEL and BSCALE values.

**C)** Problems encountered:

No major problems or difficulties, just did a lot of reading.

**D)** Future Work/Applications:

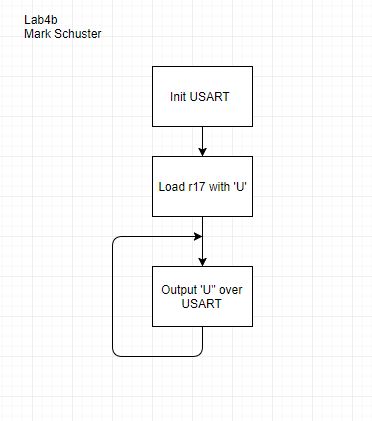
I really enjoyed using playing with serial and I have a fair amount of experience using it for to transmit JSON data between systems.

**E)** Schematics:

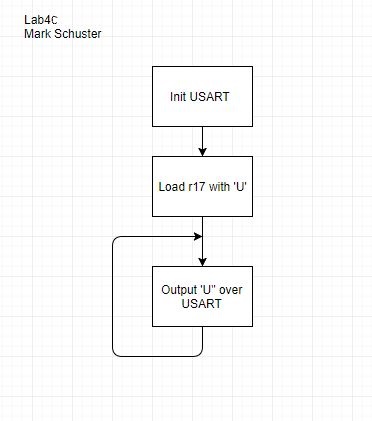
Not applicable for this lab.

**F)** Pseudocode/Flowcharts:

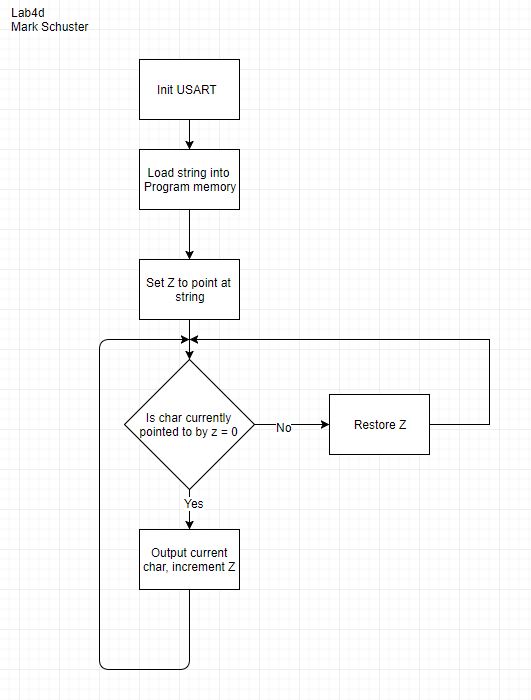
Lab4b flow diagram:



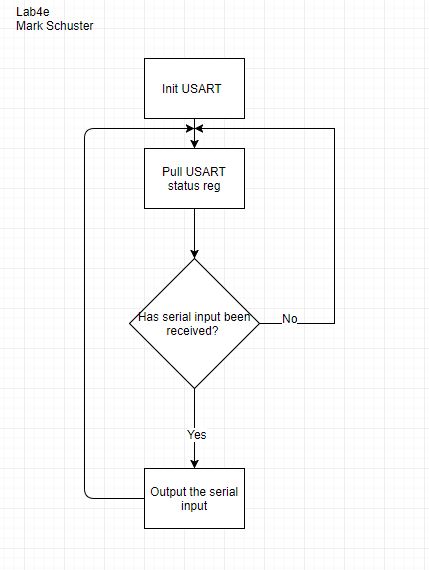
Lab4c flow diagram:



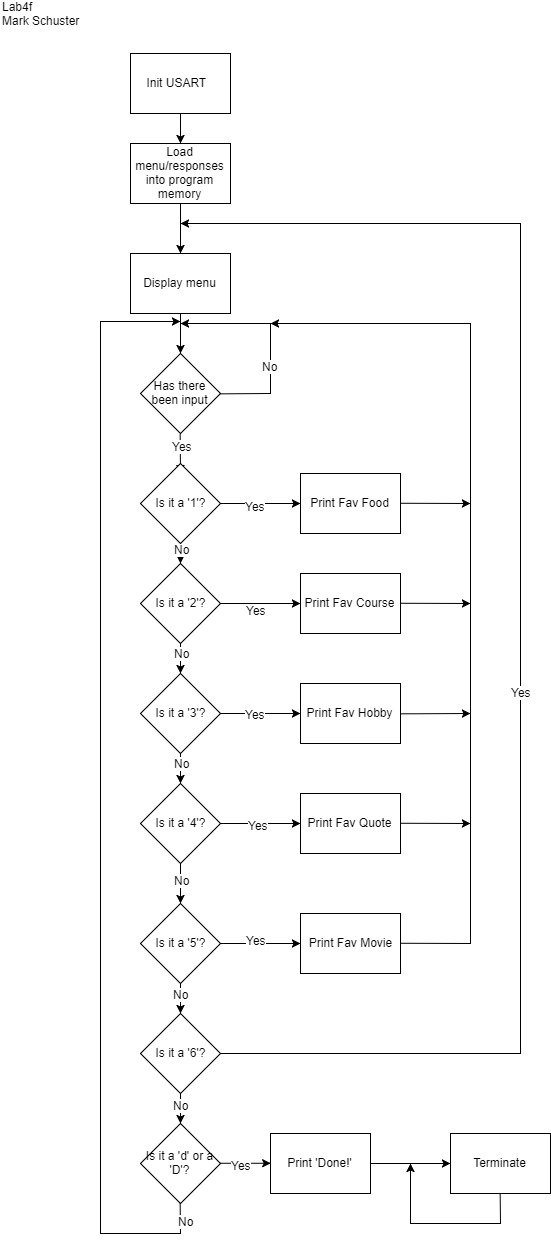
Lab4d flow diagram:



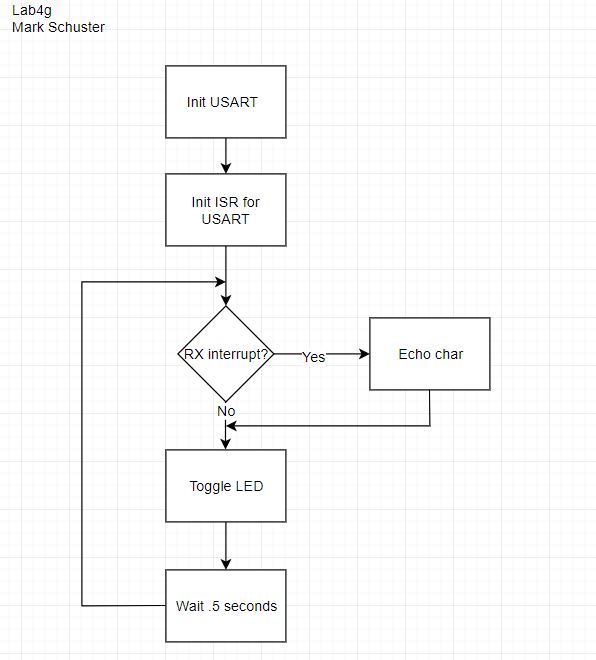
Lab4e flow diagram:



Lab4f flow diagram:



Lab4g flow diagram:



**G)** Program Code:

Lab4b.asm:

; Lab 4 part B

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Using UART to output chars

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val for the stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

ldi r17, 'U' ; Load the USART char to be outputted.

main:

rcall OUT\_CHAR ; Output a char.

rjmp main ; Endlessly loop.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTD\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTD\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTD0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTD0\_BAUDCTRLB, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTD0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTD0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTD0\_DATA, r17

checkTx:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

Lab4c.asm:

; Lab 4 part C

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Output UART to monitoring.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val for the stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

ldi r17, 'U' ; Load the USART char to be outputted.

main:

rcall OUT\_CHAR ; Output a char.

rjmp main ; Endlessly loop.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTC\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTC\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTC0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTC0\_BAUDCTRLB, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTC0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTC0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTC0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTC0\_DATA, r17

checkTx:

lds r16, USARTC0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

Lab4d.asm:

; Lab 4 part D

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Using to UART to output strings.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val fot eh stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

; ~~~ Used in MAIN ~~~ ;

.equ STRING\_ADDR = 0x200

.equ CR = 0x0D

.equ LF = 0x0A

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org STRING\_ADDR

.db "Peanut Butter Chocolate flavor"

.db CR, LF, 0x00, 0x00

.org 0x300

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

ldi ZH, byte3(STRING\_ADDR << 1)

out CPU\_RAMPZ, ZL

ldi ZH, byte2(STRING\_ADDR << 1)

ldi ZL, byte1(STRING\_ADDR << 1)

main:

rcall OUT\_STRING ; Output a string.

rjmp main ; Endlessly loop.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTD\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTD\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTD0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTD0\_BAUDCTRLB, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTD0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTD0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTD0\_DATA, r17

checkTx:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_STRING

; Send the character stored in Z over UART

; Inputs: Z - String input

; Outputs: None

; Affected: r16, Z

OUT\_STRING:

push r16 ; Preserve r16 and the Z pointer.

mov r16, ZL

push r16

mov r16, ZH

push r16

lds r16, CPU\_RAMPZ

push r16

stringLoop:

lpm r17, Z+

cpi r17, 0x00

breq stringRet

rcall OUT\_CHAR

rjmp stringLoop

stringRet:

pop r16 ; Restore r16 and the Z pointer.

mov ZL, r16

out CPU\_RAMPZ, r16

pop r16

mov ZH, r16

pop r16

mov ZL, r16

pop r16

ret ; Return.

Lab4e.asm:

; Lab 4 part E

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: USing to UART to echo chars.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val fot eh stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

main:

rcall IN\_CHAR ; Input a char.

rcall OUT\_CHAR ; Output a char.

rjmp main ; Endlessly loop.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTD\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTD\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTD0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTD0\_BAUDCTRLB, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTD0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTD0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTD0\_DATA, r17

checkTx:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: IN\_CHAR

; Send the character stored in r17 over UART

; Inputs: None

; Outputs: r17 - The value of the char taken in.

; Affected: r16, r17

IN\_CHAR:

push r16 ; Preserve the value of r16.

checkRx:

lds r16, USARTD0\_STATUS ; Pull the data status flag and wait

sbrs r16, 7 ; until it is ready to receive again.

rjmp checkRx

lds r17, USARTD0\_DATA

pop r16 ; Restore r16.

ret ; return.

Lab4f.asm:

; Lab 4 part F

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Using to UART to a menu and take input.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val fot eh stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

; ~~~ Used in MAIN ~~~ ;

.equ STRING\_ADDR = 0x200

.equ CR = 0x0D

.equ LF = 0x0A

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org STRING\_ADDR

header:

.db "Mark's favorite:", CR, LF, 0x00

food:

.db "1. Food", CR, LF, 0x00

course:

.db "2. UF Course", CR, LF, 0x00, 0x00

hobby:

.db "3. Hobby", CR, LF, 0x00, 0x00

quote:

.db "4. Quote", CR, LF, 0x00, 0x00

movie:

.db "5. Movie", CR, LF, 0x00, 0x00

menu:

.db "6. Re-display menu", CR, LF, 0x00, 0x00

done:

.db "D: Done", CR, LF, 0x00

foodRsp:

.db "Mark's favorite food is italian.", CR, LF, 0x00, 0x00

courseRsp:

.db "Mark's favorite UF course is EEL3701.", CR, LF, 0x00

hobbyRsp:

.db "Mark's favorite hobby is finance.", CR, LF, 0x00

quoteRsp:

.db "Mark's favorite quote is ", '"', "The more you learn, the more you earn.", '"', " - Warren Buffet.", CR, LF, 0x00, 0x00

movieRsp:

.db "Mark's favorite movie is The Pursuit of Happiness.", CR, LF, 0x00, 0x00

doneRsp:

.db "Done!", CR, LF, 0x00

.org 0x400

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

ldi ZH, byte3(STRING\_ADDR << 1)

out CPU\_RAMPZ, ZL

ldi ZH, byte2(STRING\_ADDR << 1)

ldi ZL, byte1(STRING\_ADDR << 1)

rcall OUT\_MENU ; Output Menu

main:

rcall IN\_CHAR ; Take input

cpi r17, '1' ; Switch statement for each case.

breq choice1

cpi r17, '2'

breq choice2

cpi r17, '3'

breq choice3

cpi r17, '4'

breq choice4

cpi r17, '5'

breq choice5

cpi r17, '6'

breq choice6

cpi r17, 'd'

breq doneTerm

cpi r17, 'D'

breq doneTerm

rjmp main ; Endlessly loop.

choice1: ; Output each response depending on the input.

ldi ZH, high(foodRsp << 1)

ldi ZL, low(foodRsp << 1)

rcall OUT\_STRING

rjmp main

choice2:

ldi ZH, high(courseRsp << 1)

ldi ZL, low(courseRsp << 1)

rcall OUT\_STRING

rjmp main

choice3:

ldi ZH, high(hobbyRsp << 1)

ldi ZL, low(hobbyRsp << 1)

rcall OUT\_STRING

rjmp main

choice4:

ldi ZH, high(quoteRsp << 1)

ldi ZL, low(quoteRsp << 1)

rcall OUT\_STRING

rjmp main

choice5:

ldi ZH, high(movieRsp << 1)

ldi ZL, low(movieRsp << 1)

rcall OUT\_STRING

rjmp main

choice6:

rcall OUT\_MENU

rjmp main

doneTerm:

ldi ZH, high(doneRsp << 1) ; If 'd' || 'D', print response and terminate.

ldi ZL, low(doneRsp << 1)

rcall OUT\_STRING

end:

rjmp end

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTD\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTD\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTD0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTD0\_BAUDCTRLB, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTD0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTD0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTD0\_DATA, r17

checkTx:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: IN\_CHAR

; Send the character stored in r17 over UART

; Inputs: None

; Outputs: r17 - The value of the char taken in.

; Affected: r16, r17

IN\_CHAR:

push r16 ; Preserve the value of r16.

checkRx:

lds r16, USARTD0\_STATUS ; Pull the data status flag and wait

sbrs r16, 7 ; until it is ready to receive again.

rjmp checkRx

lds r17, USARTD0\_DATA

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_STRING

; Send the character stored in Z over UART

; Inputs: Z - String input

; Outputs: None

; Affected: r16, Z

OUT\_STRING:

push r16 ; Preserve r16 and the Z pointer.

mov r16, ZL

push r16

mov r16, ZH

push r16

lds r16, CPU\_RAMPZ

push r16

stringLoop:

lpm r17, Z+

cpi r17, 0x00

breq stringRet

rcall OUT\_CHAR

rjmp stringLoop

stringRet:

pop r16 ; Restore r16 and the Z pointer.

mov ZL, r16

out CPU\_RAMPZ, r16

pop r16

mov ZH, r16

pop r16

mov ZL, r16

pop r16

ret ; Return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_MENU

; Display the menu over UART

; Inputs: None

; Outputs: None

; Affected: r16, Z

OUT\_MENU:

push r16 ; Preserve r16 and the Z pointer.

mov r16, ZL ; Load Z with the starting value of

push r16 ; the table data.

mov r16, ZH

push r16

lds r16, CPU\_RAMPZ

push r16

ldi ZL, low(header << 1) ; Load each line and output it over UART.

ldi ZH, high(header << 1)

rcall OUT\_STRING

ldi ZL, low(food << 1)

ldi ZH, high(food << 1)

rcall OUT\_STRING

ldi ZL, low(course << 1)

ldi ZH, high(course << 1)

rcall OUT\_STRING

ldi ZL, low(hobby << 1)

ldi ZH, high(hobby << 1)

rcall OUT\_STRING

ldi ZL, low(quote << 1)

ldi ZH, high(quote << 1)

rcall OUT\_STRING

ldi ZL, low(movie << 1)

ldi ZH, high(movie << 1)

rcall OUT\_STRING

ldi ZL, low(menu << 1)

ldi ZH, high(menu << 1)

rcall OUT\_STRING

ldi ZL, low(done << 1)

ldi ZH, high(done << 1)

rcall OUT\_STRING

pop r16 ; Restore r16 and the Z pointer.

mov ZL, r16

out CPU\_RAMPZ, r16

pop r16

mov ZH, r16

pop r16

mov ZL, r16

pop r16

ret ; Return.

Lab4g.asm:

; Lab 4 part G

; Name: Mark L. Schuster

; Section #: 1540

; TA Name: Christopher Crary

; Description: Using to UART to echo chars driven by interrupts.

.nolist ; Included for fun.

.include "ATxmega128A1Udef.inc" ;

.list ;

; ~~~ General ~~~ ;

.equ STACKINIT = 0x3FFF ; Init val of the stack ptr.

; ~~~ Used in USE32MHzCLK ~~~ ;

.def clkPrescaler = r17 ; Denotes the input for the USE32MHzCLK input. Will hold the prescaler value.

.equ CLKEN = 0b0010 ; Enables the 32Mhz CLK.

.equ IOREG = 0xD8 ; The value that sets the CPU\_CCP reg to 'IOREG' mode.

.equ CLKPS = 0b00000000 ; Value that sets Prescaler A to 4.

.equ CLKSEL = 1 ; Value to select the 32MHz CLK.

.equ CLKOUT = 0b00000001 ; Value to output the CLK signal to port C.

; ~~~ Used in USART\_INIT ~~~ ;

.equ CRTLA\_INIT = 0b00010000

.equ CRTLB\_INIT = 0b00011000 ; En TX & RX, STD CLK, Non-MPCM, Non-9 bit data.

.equ CRTLC\_INIT = 0b00110011 ; Async, Odd parity, 1 stop bit, 8 data bits.

.equ BSEL\_VAL = 107 ; 107

.equ BSCALE\_VAL = 0b1011 ; -5 2's comp

.equ USART\_BAUDCA\_VAL = low(BSEL\_VAL) ; Value to init the USART BUADA reg.

.equ USART\_BAUDCB\_VAL = (high(BSEL\_VAL)>>4) | (BSCALE\_VAL << 4) ; Value to init the USART BUADB reg.

; ~~~ Used in DELAY\_500ms ~~~ ;

.equ TC\_SEL = 0b0111 ; Value to set the prescaler of the TC to be 1024 time the period of the system CLK.

.equ TC\_PER = 0x3D09 ; Value of the TC period.

.equ TC\_DISABLE = 0b0000

; ~~~ Used in MAIN ~~~ ;

.equ RGB\_GREEN\_DIR\_SET = 0b00100000 ; Value to set the direction of the blue RGB LED to output.

.org 0x0000

rjmp init ; Start at 0x0000 and jump to program.

.org 0x200

init:

ldi clkPrescaler, CLKPS ; Standard inits of the CLK, the RGB, stack ptr.

rcall USE32MHzCLK

ldi XL, low(STACKINIT)

out CPU\_SPL, XL

ldi XL, high(STACKINIT)

out CPU\_SPH, XL

rcall USART\_INIT ; Init USART on port D.

ldi r16, PMIC\_LOLVLEN\_bm ; Enable low priority interrupts.

sts PMIC\_CTRL, r16

sei

ldi r16, RGB\_GREEN\_DIR\_SET ; Set the blue RGB LED as output.

sts PORTD\_DIRSET, r16

ldi r18, RGB\_GREEN\_DIR\_SET

main:

sts PORTD\_OUTTGL, r18 ; Toggle backpack LEDs and loop infinitely.

rcall DELAY\_500ms

rjmp main ; Endlessly loop.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INTERUPT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: RX\_ISR

; Sets up an interupt to be triggered by the S1 button to

; increment a count and output it to the LEDs.

; Inputs: None

; Outputs: None

; Affected: r16, r17

RX\_ISR:

push r17

push r16

lds r17, USARTD0\_DATA

checkDre1:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre1

sts USARTD0\_DATA, r17

checkTx1:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx1

pop r16

pop r17

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USE32MHzCLK

; Sets the external 32MHz as the active clock for the device

; Inputs: r17 as the desired prescaler for the clock

; Outputs: None

; Affected: r16, r17

USE32MHzCLK:

push r16 ; Preserve the values of r16, r17.

push r17 ;

ldi r16, CLKEN ; Load the CLK enable value and store it in the CLK control.

sts OSC\_CTRL, r16 ;

checkReady:

lds r16, OSC\_STATUS ; This section pulls the oscillator status reg and constantly

andi r16, CLKEN ; checks if the 32Mhz CLK is ready yet.

cpi r16, CLKEN ;

breq clockSel ; If it is move on, if not loop continuously.

rjmp checkReady ;

clockSel:

ldi r16, IOREG ; Write 'IOREG' to the CPU\_CCP to allow the CLK Prescaler

sts CPU\_CCP, r16 ; to be written to.

sts CLK\_PSCTRL, clkPrescaler ;

sts CPU\_CCP, r16 ; Write 'IOREG' to the CPU\_CCP to allow the CLK Control

ldi r16, CLKSEL ; to be set to output the 32 MHz.

sts CLK\_CTRL, r16 ;

pop r17 ; Restore the values of r16 and r17.

pop r16 ;

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: USART\_INIT

; Initializes the UART interface

; Inputs: None

; Outputs: None

; Affected: r16

USART\_INIT:

push r16 ; Preserve the values of r16.

ldi r16, TC\_DISABLE ; Break from the loop and disable the TC.

sts TCC1\_CTRLA, r16

ldi r16, USART\_TXEN\_bm ; Set the direction of the TX and RX pins.

sts PORTD\_DIRSET, r16

ldi r16, USART\_RXEN\_bm

sts PORTD\_DIRCLR, r16

ldi r16, USART\_BAUDCA\_VAL ; Set the BAUD rate.

sts USARTD0\_BAUDCTRLA, r16

ldi r16, USART\_BAUDCB\_VAL

sts USARTD0\_BAUDCTRLB, r16

ldi r16, CRTLA\_INIT ; Set the interrupt levels.

sts USARTD0\_CTRLA, r16

ldi r16, CRTLC\_INIT ; Set the USART mode and config.

sts USARTD0\_CTRLC, r16

ldi r16, CRTLB\_INIT ; Enable the pins.

sts USARTD0\_CTRLB, r16

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: OUT\_CHAR

; Send the character stored in r17 over UART

; Inputs: r17 - Character input

; Outputs: None

; Affected: r16, r17

OUT\_CHAR:

push r16 ; Preserve the values of r16, r17.

checkDre:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 5 ; until it is ready to transmit again.

rjmp checkDre

sts USARTD0\_DATA, r17

checkTx:

lds r16, USARTD0\_STATUS ; Pull the TX status flag and wait

sbrs r16, 6 ; until it is ready to transmit again.

rjmp checkTx

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: IN\_CHAR

; Send the character stored in r17 over UART

; Inputs: None

; Outputs: r17 - The value of the char taken in.

; Affected: r16, r17

IN\_CHAR:

push r16 ; Preserve the value of r16.

checkRx:

lds r16, USARTD0\_STATUS ; Pull the data status flag and wait

sbrs r16, 7 ; until it is ready to receive again.

rjmp checkRx

lds r17, USARTD0\_DATA

pop r16 ; Restore r16.

ret ; return.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SUBROUTINES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Subroutine Name: DELAY\_500ms

; Delays 500ms

; Inputs: None

; Outputs: None

; Affected: r16

DELAY\_500ms:

push r16 ; Preserve r16.

ldi r16, TC\_SEL ; Enable the TC and set its period to be that of the system CLK.

sts TCC1\_CTRLA, r16

ldi r16, low(TC\_PER) ; Load the period of the TC into the TC's period regs.

sts TCC1\_PER, r16

ldi r16, high(TC\_PER)

sts TCC1\_PER+1, r16

DELAY\_500ms\_loop: ; Loop checking the TC's interrupt flags until

lds r16, TCC1\_INTFLAGS ; the overflow flag is set.

sbrs r16, 0

rjmp DELAY\_500ms\_loop

sts TCC1\_INTFLAGS, r16

ldi r16, TC\_DISABLE ; Break from the loop and disable the TC.

sts TCC1\_CTRLA, r16

pop r16 ; Restore r16.

ret ; Return.

.org USARTD0\_RXC\_vect

rcall RX\_ISR

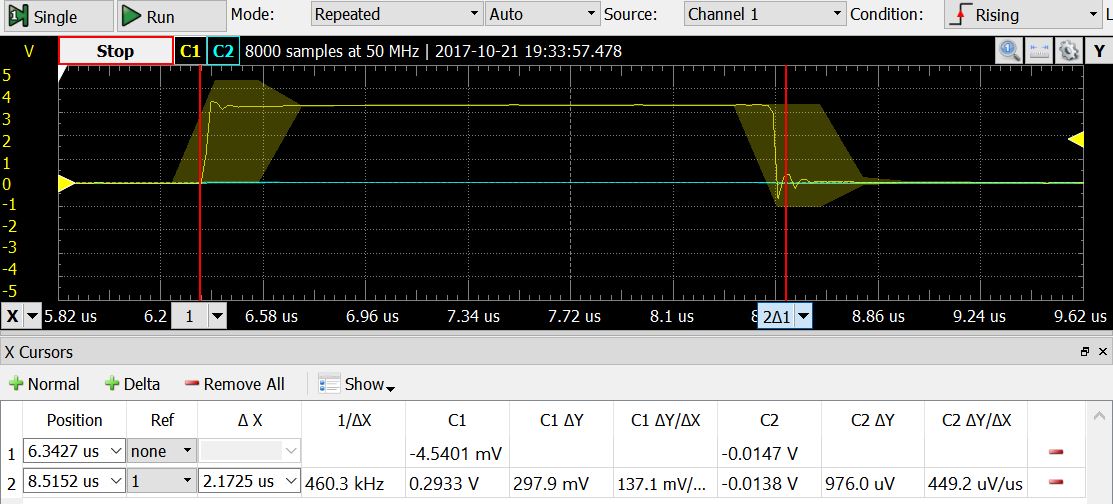
**H)** Appendix:

Files:

* Lab4.pdf
* Lab4b.asm
* Lab4c.asm
* Lab4d.asm
* Lab4e.asm
* Lab4f.asm
* Lab4g.asm

Screenshots:

Single char:



Single frame:

