Lab 4 Summary

Schuster, Mark

Section 1540

10/21/2017

B) Pre-Lab Answers:

1. **Pins 3 & 4 of port D.**
2. **2Mbps** $0 > BSCALE \geq -7, \rightarrow \frac{1}{2^{BSCALE}}\left(\frac{32*10^6}{16*f_{BAUD}} - 1\right) = BSEL, vBSEL \geq 0 \rightarrow f_{BAUD} = 2MHz \rightarrow \frac{1}{2^{BSCALE}}\left(\frac{32*10^6}{32*10^6} - 1\right) = 0$, 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.
5.
    a. `USARTD0_DATA – Holds the transmission and received data`
    b. `USARTD0_STATUS – Holds the status bits, namely RX, TX, & DR flags.`
    c. `USARTD0_CTRLA – Used to enable and set the priority levels of the interrupt levels for the port's USART.`
    d. `USARTD0_CTRLB – Used to enable the TX & RX pins, and enable different USART modes.`
    e. `USARTD0_CTRLC - Used to set the mode, size, parity, and number of stop bits.`
    f. 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:

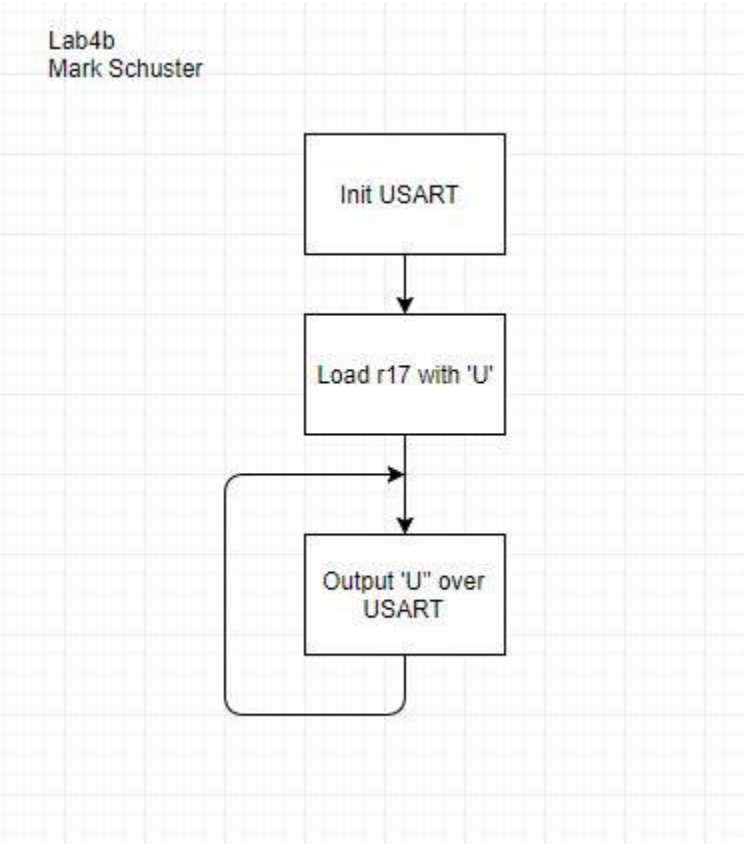
I really enjoyed using playing with serial and I have a fair amount of experience using it for to transmit JSON data between systems.

E) Schematics:

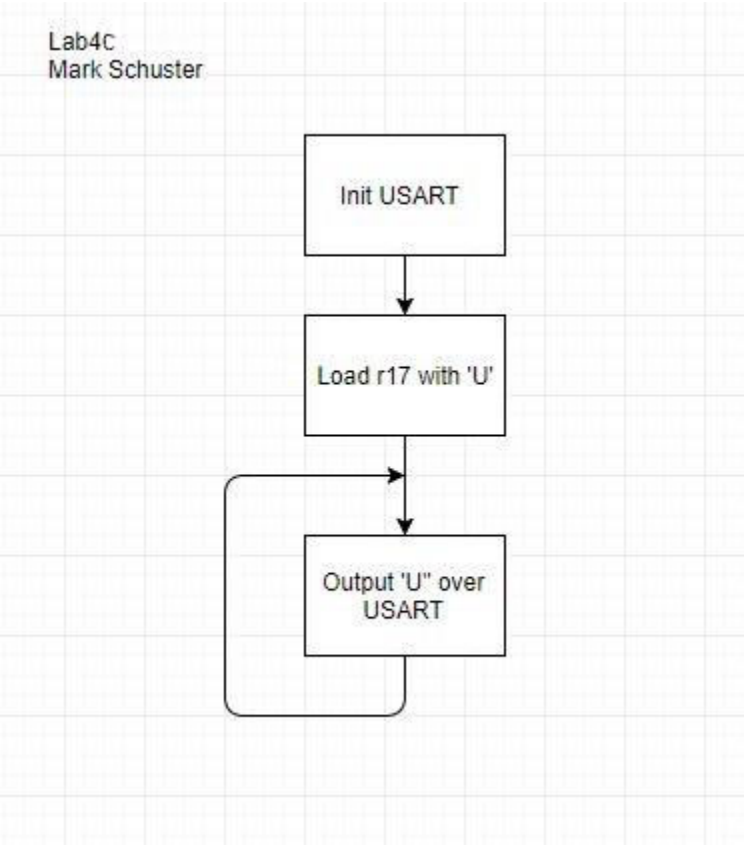
Not applicable for this lab.

F) Pseudocode/Flowcharts:

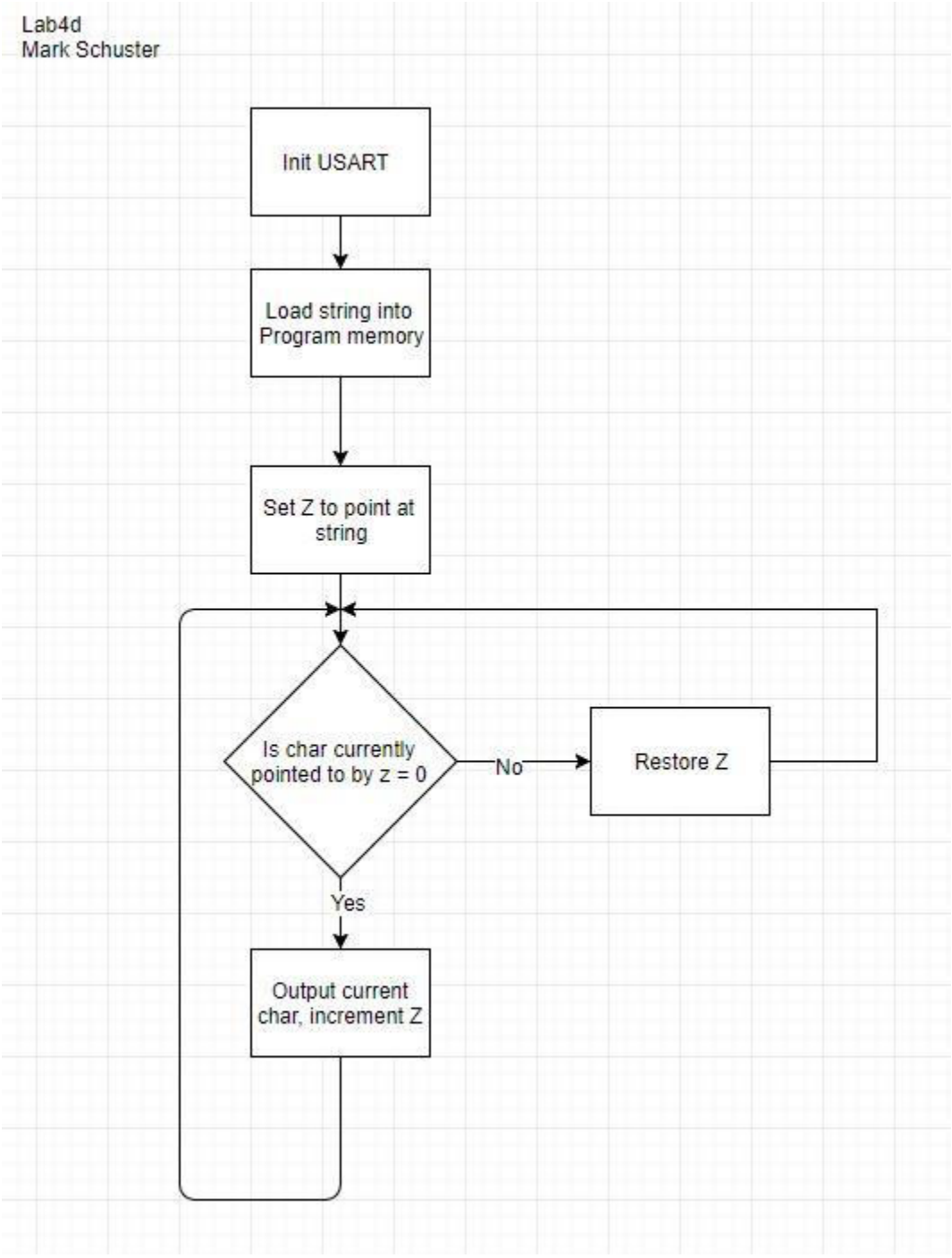
Lab4b flow diagram:



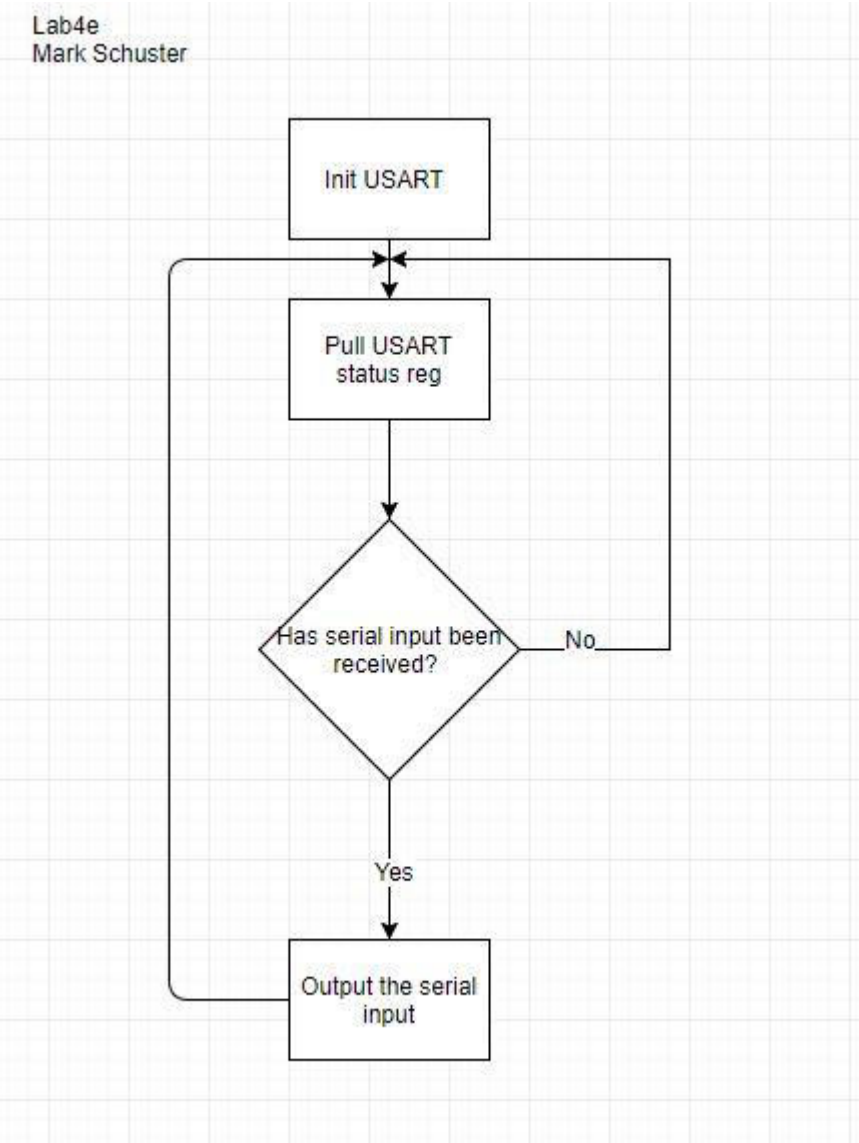
Lab4c flow diagram:



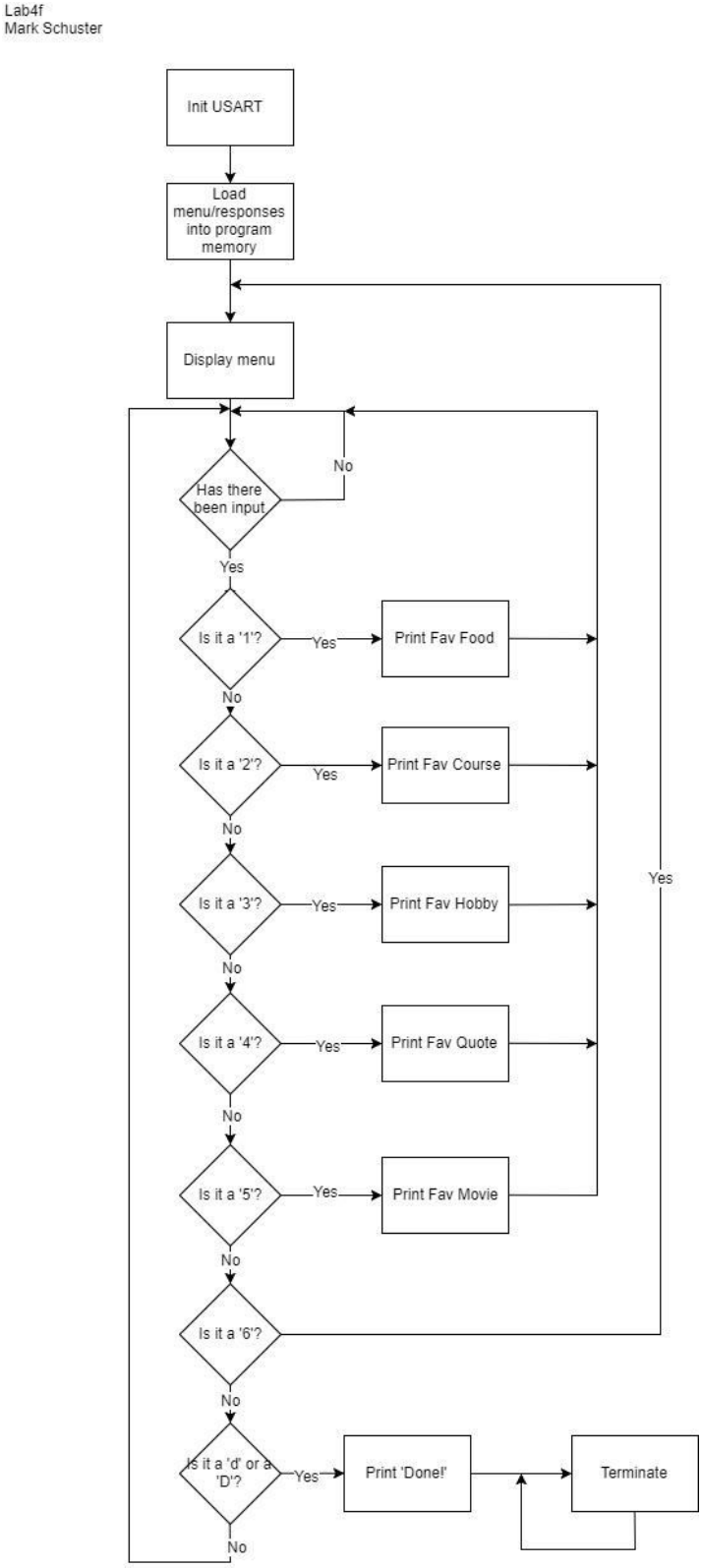
Lab4d flow diagram:



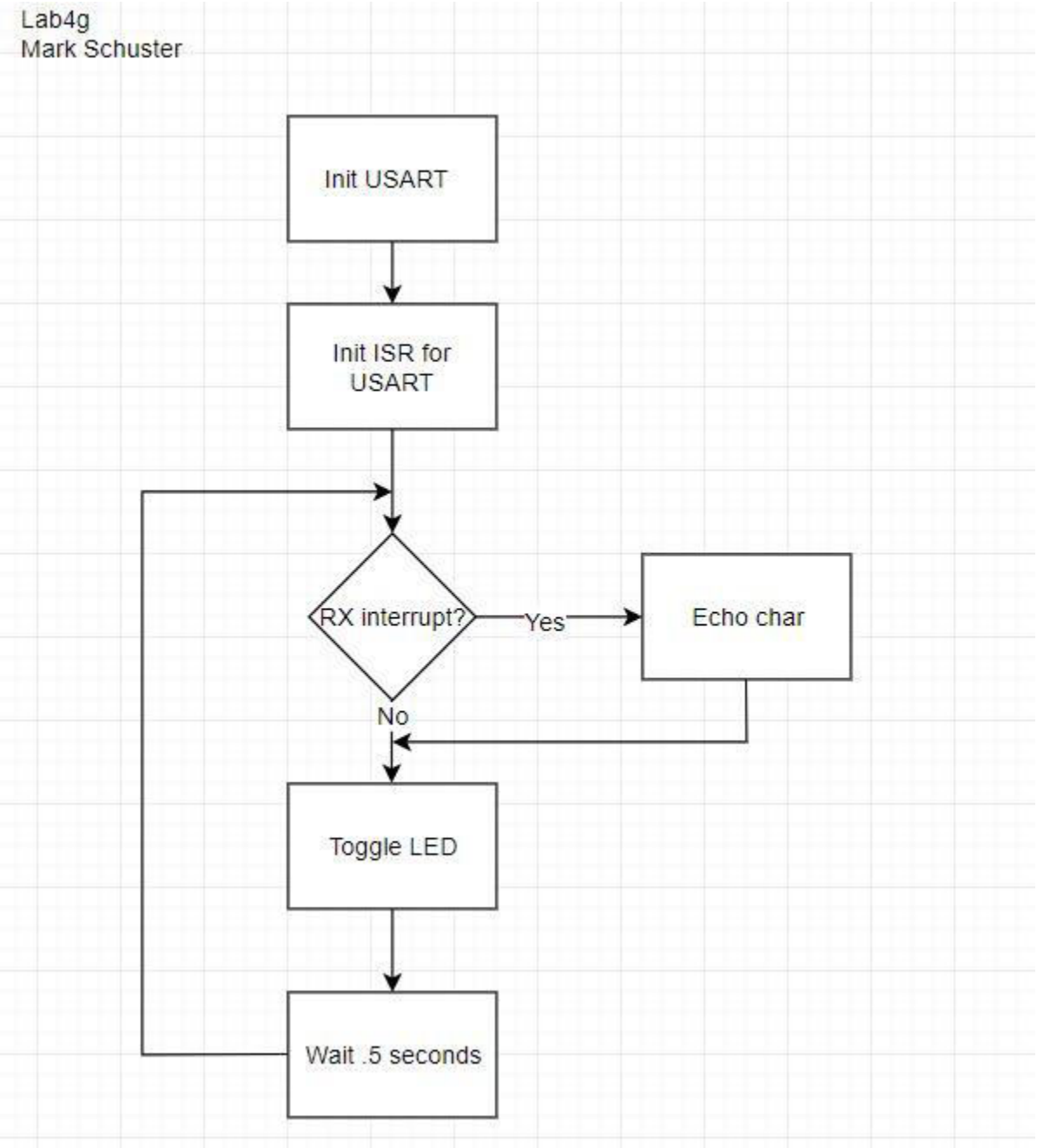
Lab4e flow diagram:



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
```

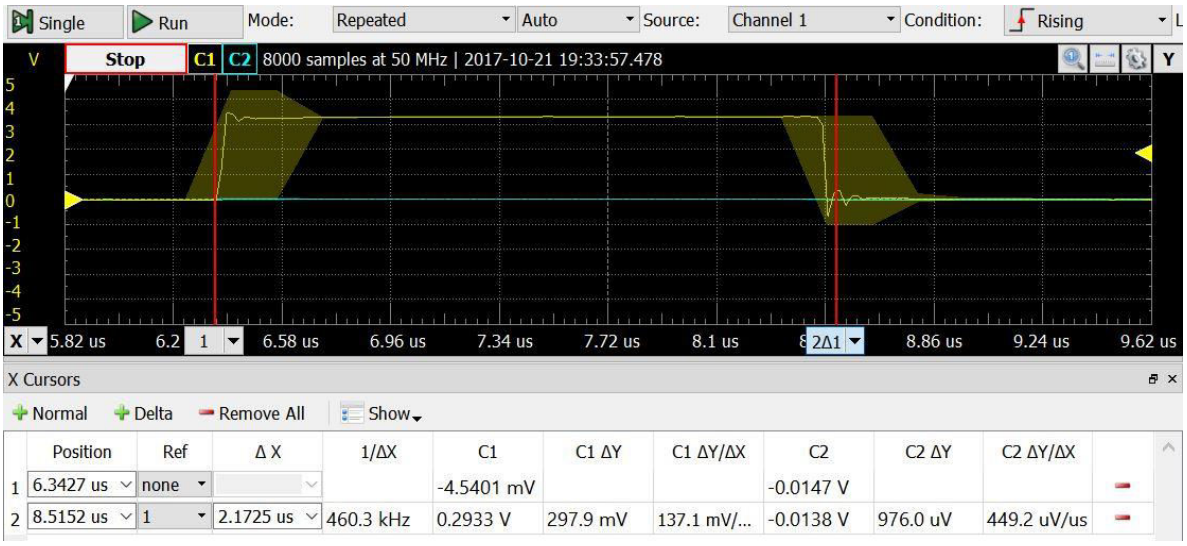
H) Appendix:

Files:

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

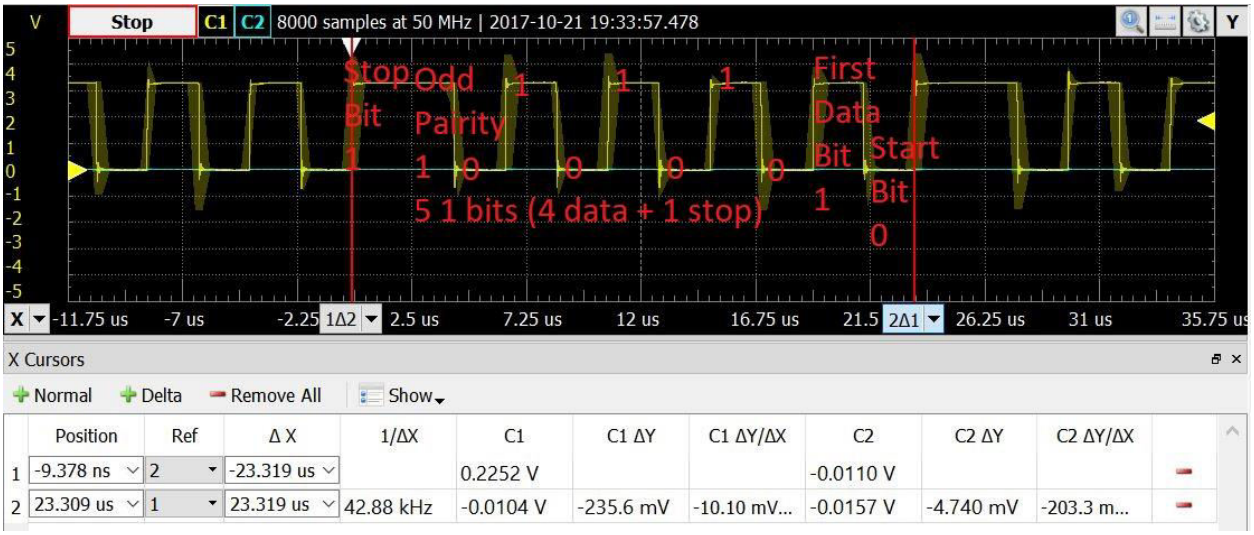
Screenshots:

Single char:



$$\frac{1}{2.1725}\mu S = 460300Hz \cong 460800Hz \rightarrow 0.1\%\ error$$

Single frame:



X Cursors

Normal | Delta | Remove All | Show

	Position	Ref	Δ X	1/ΔX	C1	C1 ΔY	C1 ΔY/ΔX	C2	C2 ΔY	C2 ΔY/ΔX
1	-9.378 ns	2	-23.319 us		0.2252 V			-0.0110 V		
2	23.309 us	1	23.319 us	42.88 kHz	-0.0104 V	-235.6 mV	-10.10 mV...	-0.0157 V	-4.740 mV	-203.3 m...