Lab: 4
Nicholas Imamshah
Section: 6957

b) Prelab Questions

1. What is the difference between synchronous and asynchronous communication?
    a. With synchronous communication, an external clock is used to synchronize sending and receiving. With asynchronous communication, special signals such as "DREIF" on the XMEGA are used to synchronize sending and receiving.
2. What is the difference between serial and parallel communication?
    a. Serial communication uses a single transmission medium, and transmits data one bit at a time. Parallel communication uses multiple transmission mediums simultaneously to transmit multiple bits at once.
3. List the XMEGA's USART registers used in your programs and briefly describe their functions.
    a. USARTp#_STATUS: Provides the current status of the Rx/Tx. Checking appropriate values allows for receiving or transmitting at the right times.
    b. USARTp#_DATA: Register to check when it is time to transmit or receive.
    c. USARTp#_CTRLA: Controls USART interrupt levels.
    d. USARTp#_CTRLB: Configures USART, enables Rx/Tx.
    e. USARTp#_CTRLC: Configures USART's communication mode, parity mode, stop bit mode, character size, and data order.
    f. USARTp#_BAUDCTRLA: Holds the lower 8 bits of the BSel.
    g. USARTp#_BAUDCTRLB: Holds the BScale and upper 4 bits of the BSel.
4. List the number of bounces from part A of this lab. How long (in ms) is your delay routine for debouncing?
    a. Variable from 0 to ~8. My delay routine is approximately 10ms.
5. What is the maximum possible baud you can use for asynchronous communication if your board runs at 2MHz? Support your answer with the values you would place in any special registers that are needed.
    a. Max Baud = 125000Hz; BSel = 0; BScale = 0

c) Problems Encountered

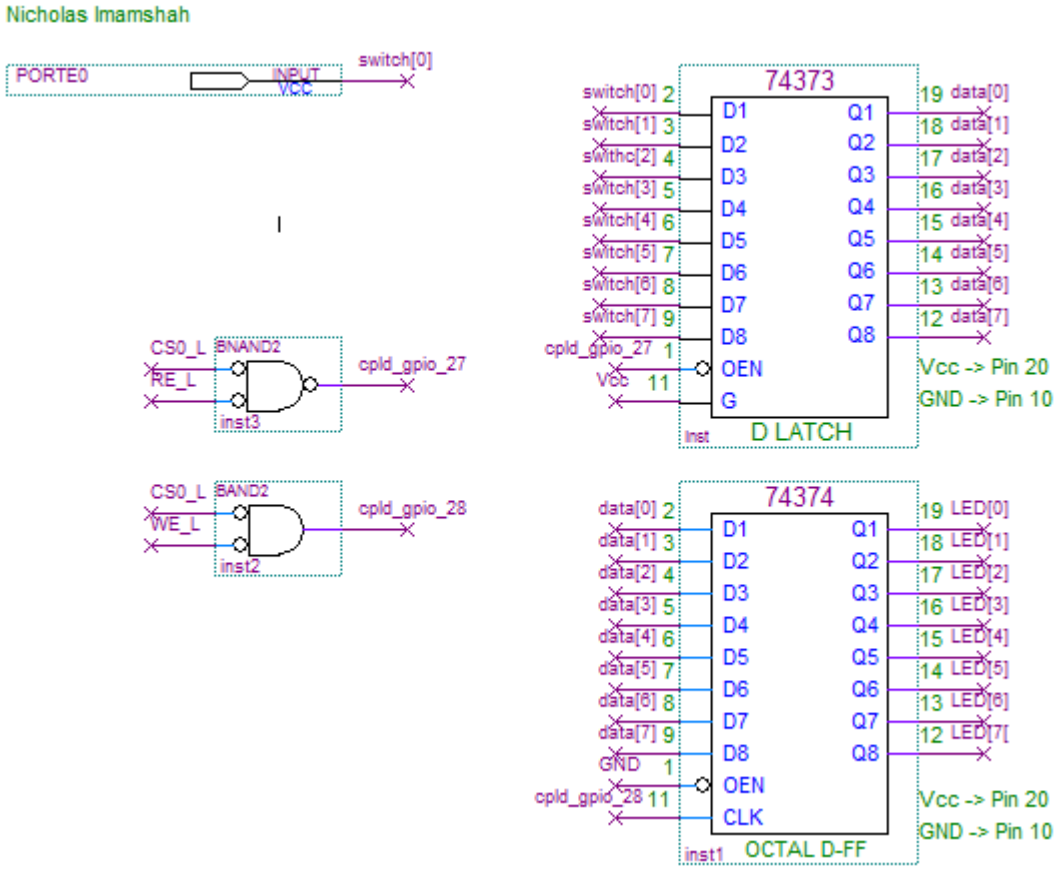
1. Initially I forgot to include the null character at the end of my strings, so I was having trouble debugging the OUT_STRING method in part E.
2. I did not understand how to verify the baud rate initially; mainly I misunderstood what was meant by the lab document when it mentioned using Port E or Port F for the USART.

d) Future Work/Applications

In the lab we used serial communication to transmit ASCII characters from our processor to our computer; this concept represents the much more general process of sending a stream of data

between devices. Such a configuration is reminiscent of a networking scenario, where there is a sending a receiving device, the receiving device constantly accepting packets of some sort from the sending device.

e) Schematics

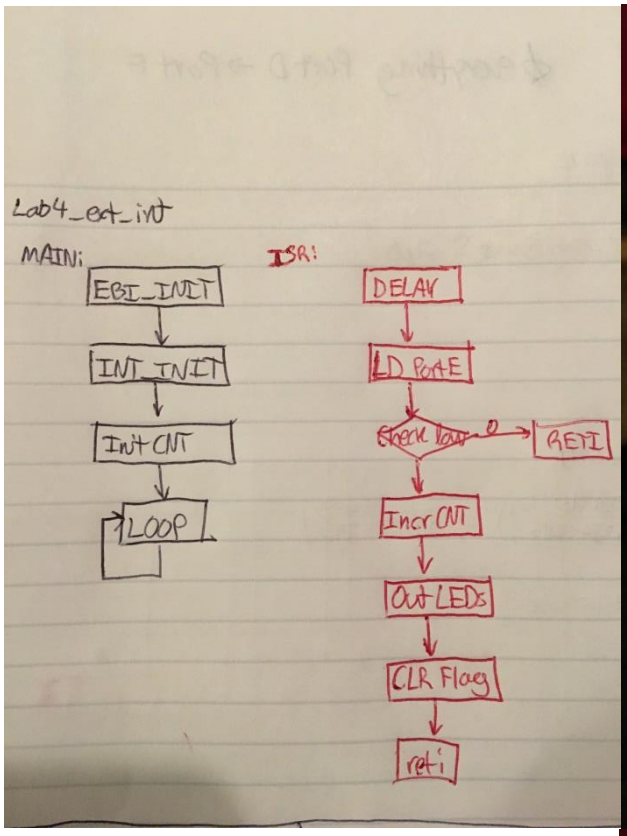


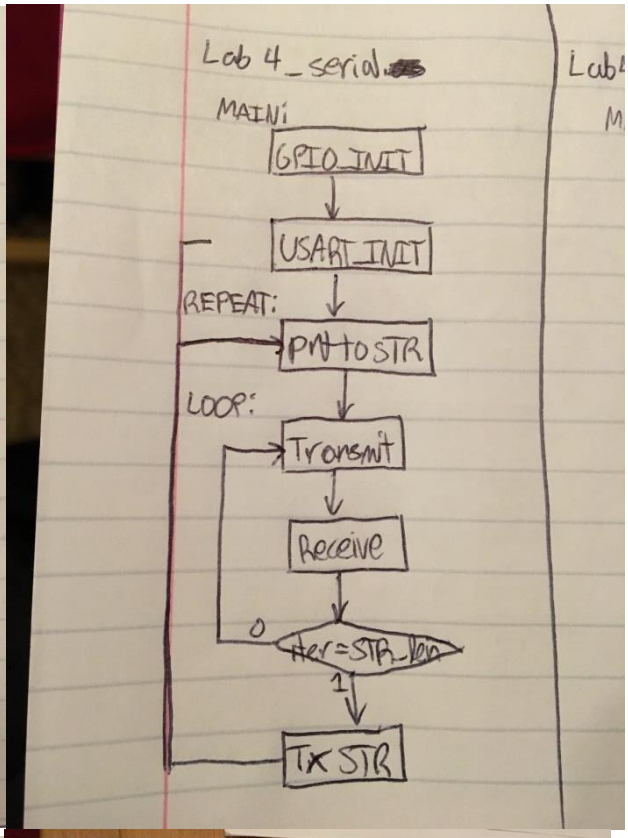
f) Decoding Logic

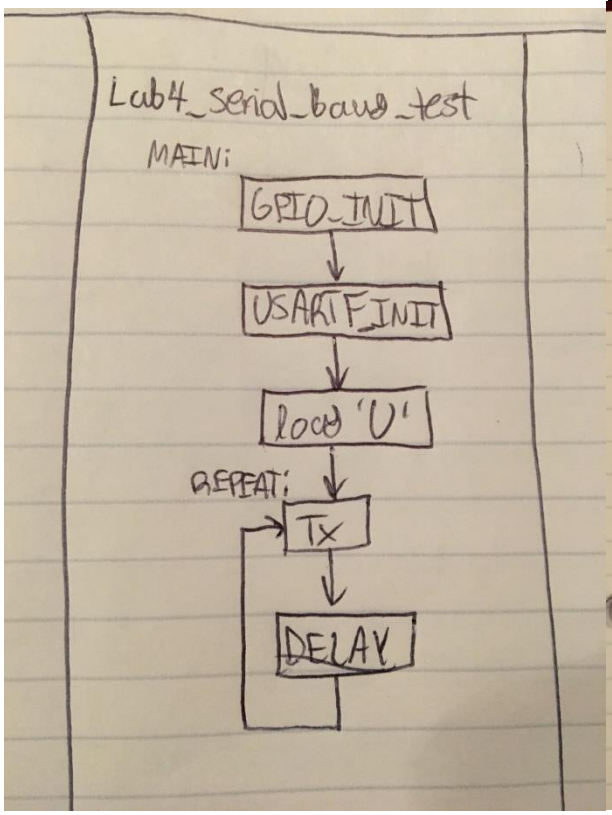
N/A

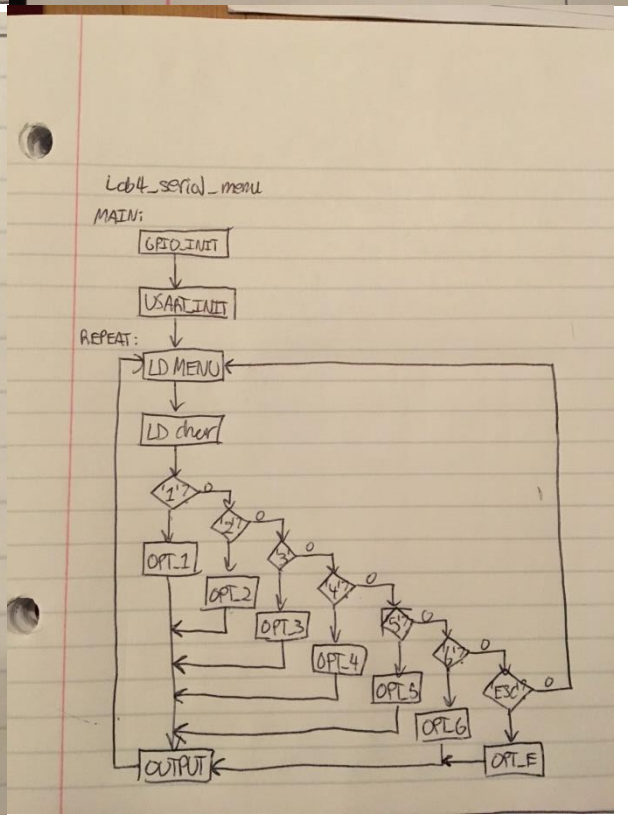
g) Pseudocode/Flowcharts

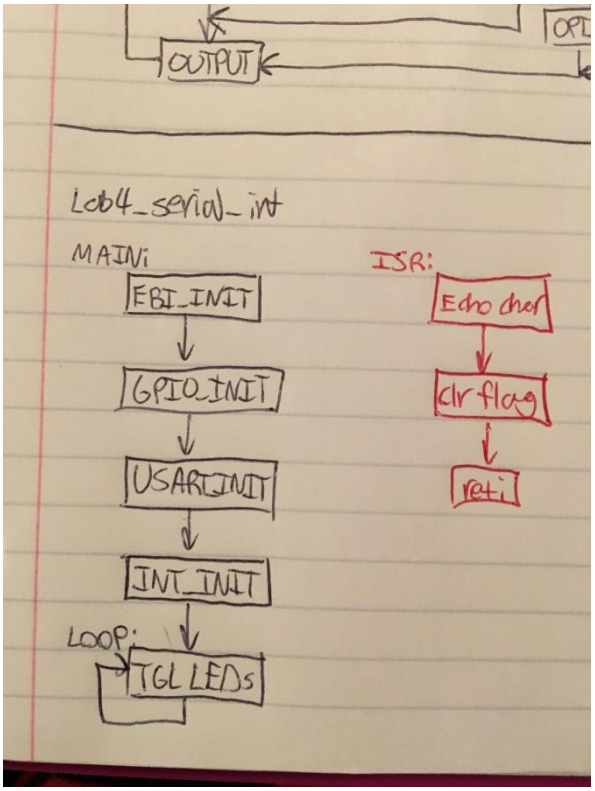
Lab: 4
Nicholas Imamshah
Section: 6957











h) Program Code

Lab4_ext_int.asm

```
/* Lab4_ext_intr.asm
 *
 * Lab 4 External Interrupt
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to configure an interrupt on the XMEGA that will watch for a falling edge.
 */

.nolist
.include "ATxmega128A1Udef.inc"
.list

.org 0x0000
		rjmp MAIN

.org PORTE_INT0_VECT			;place code at the interrupt vector for the PORTE_INT0 interrupt
		jmp ISR_LED_COUNT			;jump to our interrupt routine

;must org program at 0x0200 so it doesn't conflict with interrupt vectors that are at 0x0000-0x00FE
.org 0x0200
MAIN:
		rcall CONFIG_EBI
```

```
		nop
		rcall INIT_INTERRUPT
		nop

		ldi r18, 0				;initialize count

LOOP:
		rjmp LOOP
/*****************************************************************************
* Name:    INIT_INTERRUPT
* Purpose:  Subroutine to initialize the PortE external pin interrupt PE0 using INT0
* Inputs:  None
* Outputs:  None
* Affected: r16, PMIC_CTRL, PORTE: _INT0MASK_OUT, _DIRCLR, _INTCTRL, _PIN0CTRL
 *****************************************************************************/
INIT_INTERRUPT:
		ldi r16, 0x01				;set output to default to '1'.
		sts PORTE_OUT, r16
		sts PORTE_DIRCLR, r16		;Set external interrupt pin (PE0) as an input

		ldi r16, 0x01				;select the external interrupt as a low-level priority
		sts PORTE_INTCTRL, r16

		ldi r16, 0x01				;select PORTE_PIN0 as the interrupt source
		sts PORTE_INT0MASK, r16

		ldi r16, 0x02				;select falling edge for external interrupt
		sts PORTE_PIN0CTRL, r16

		ldi r16, 0x01				;enable low-level interrupts
		sts PMIC_CTRL, r16

		sei						;set global interrupt flag LAST!
		ret
/*****************************************************************************
* Name:    ISR_LED_COUNT
* Purpose:  Interrupt service routine to increment the count of executions on LEDs
* Inputs:  None
* Outputs:  None
* Affected: r17, r18
 *****************************************************************************/
ISR_LED_COUNT:
		ldi r16, 255
		rcall DELAY
		nop

		lds r16, PORTE_IN
		nop
		sbrc r16, 0				;if bit0 is cleared, then continue
		reti					;else, we got a rising edge
```

```
		inc r18					;inc each time ISR runs
		st X, r18				;store to LEDs
		ldi r17, 0x01
		sts PORTE_INTFLAGS, r17	;clear the PORTE_INTFLAGS
		reti					;return from the interrupt service routine

DELAY:
		ldi r17, 50
		rcall DELAY_INNER
		dec r16
		nop
		brne DELAY
		ret
DELAY_INNER:
		dec r17
		nop
		brne DELAY_INNER
		ret
/*****************************************************************************
* Name:    CONFIG_EBI
* Purpose:  Subroutine to initialize Ports H,J,K/EBI for additional Input/Output ports
* Inputs:   None
* Outputs:  None
* Affected: R16, X, Z, EBI_CTRL, PORTH,J,K,
 *****************************************************************************/
.set PORT = 0x288000
.set PORT_END = 0x289FFF

CONFIG_EBI:
		ldi r16, 0x17			;Configure PORTH bits 4, 2, 1, and 0 as outputs.
		sts PORTH_DIRSET, r16	;	These are the CS0(L), ALE1(H), RE(L), and WE(L) outputs.
								;		(CS0 is bit 4; ALE1 is bit 2; RE is bit 1; WE is bit 0)

		ldi r16, 0x13			;Default CS0(L), RE(L), and WE(L) to H = false.
		sts PORTH_OUTSET, r16	;	ALE defaults to 0 = L = false.

		ldi r16, 0xFF			;Set all PORTK pins (A15-A0) to be outputs.
		sts PORTK_DIRSET, r16

		ldi r16, 0xFF			;Set all PORTJ pins (D7-D0) to be outputs.
		sts PORTJ_DIRSET, r16

		ldi r16, 0x01			;Store 0x41 in EBI_CTRL reg to select 3 port EBI(H,J,K)
		sts EBI_CTRL, r16		;	mode and SRAM ALE1 mode

;Initialize the Z pointer to point to the base address for CS0 in memory
		ldi ZH, high(EBI_CS0_BASEADDR)
		ldi ZL, low(EBI_CS0_BASEADDR)

;Load the middle byte (A15:8) of the three byte addr into a reg and store it as the
;		LOW byte of the Base Address, BASEADDRL.
```

```
		ldi r16, byte2(PORT)
		st Z+, r16

;Load the highest byte (A23:16) of the three byte addr into a reg and store it as the
;		HIGH byte of the Base Address, BASEADDRH.
		ldi r16, byte3(PORT)
		st Z, r16

		ldi r16, 0x15				;Set to 8KB CS space and turn on SRAM mode, 0x288000 - 0x289FFF
		sts EBI_CS0_CTRLA, r16

;Steps for using the port expansion
		ldi r16, byte3(PORT)		;initialize a pointer to point to the base addr of the PORT
		sts CPU_RAMPX, r16			;use the CPU_RAMPX reg to set the third byte of the pointer

		ldi XH, high(PORT)		;set the middle (XH) and low (XL) bytes of the pointer as usual
		ldi XL, low(PORT)

		ret
```

Lab4_serial.asm

```
/* Lab4_serial.asm
 *
 * Lab 4 XMEGA USART System
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to configure the XMEGA USART for communication with a terminal program.
 */

.nolist
.include "ATxmega128A1Udef.inc"
.list

.equ BSel = 289
.equ BScale = -7

.equ Prompt = '?'

.equ StringLength = 10
.dseg
.org 0x2000                                                ;where inputs will be stored, outputs written to
STRING:  .byte StringLength

.cseg
.org 0x0000
		rjmp MAIN

.org 0x0200
MAIN:
		rcall GPIO_INIT
		rcall USART_INIT
```

```
REPEAT:
        ldi r16, Prompt                         ;load Prompt character
        ldi r17, StringLength
        ldi ZL, low(STRING)
        ldi ZH, high(STRING)

LOOP:
        rcall OUT_CHAR                          ;echo character

        rcall IN_CHAR

        st Z+, r16                              ;store input char
        dec r17
        brne LOOP
        rcall OUT_CHAR                          ;echo last entered char

        ldi r16, '!'
        rcall OUT_CHAR                          ;signify end of string
        rcall OUT_STRING

        rjmp REPEAT
/*****************************************************************************
* Name:    USART_INIT
* Purpose:  Subroutine to initialize the XMEGA USART system
* Inputs:   None
* Outputs:  None
* Affected: r16, USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA, USARTD0_BAUDCTRLB
 *****************************************************************************/
 USART_INIT:
        ldi r16, 0x18
        sts USARTD0_CTRLB, r16          ;turn on TXEN and RXEN lines

        ldi r16, 0x23
        sts USARTD0_CTRLC, r16          ;Parity = even, 8 bit frame, 1 stop bit

        ldi r16, (BSel & 0xFF)          ;select only lower 8 bits of BSel
        sts USARTD0_BAUDCTRLA, r16      ;set BAUDCTRLA to lower 8 bits of BSel

        ldi r16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)
        sts USARTD0_BAUDCTRLB, r16      ;set BAUDCTRLB to BScale | Bsel.
                                                        ;       Lower 4 bits are upper 4 bits of BSel.
                                                        ;       Upper 4 bits are upper 4 bits of BScale.
        ret
/*****************************************************************************
* Name:    GPIO_INIT
* Purpose:  Subroutine to initialize the XMEGA GPIO for use with the USART System
* Inputs:   None
* Outputs:  None
* Affected: r16, PORTD_DIR, PORTD_OUT, PORTQ_DIR, PORTQ_OUT
 *****************************************************************************/
```

```
GPIO_INIT:
        ldi r16, 0x08
        sts PORTD_DIRSET, r16           ;set PORTD_PIN3 as output for TX pin of USARTD0
        sts PORTD_OUTSET, r16           ;set the TX line to default to '1'

        ldi r16, 0x04
        sts PORTD_DIRCLR, r16           ;set RX pin for input

        ldi r16, 0xA
        sts PORTQ_DIRSET, r16           ;set pins 1 and 3 of PORTQ to output
        sts PORTQ_OUTCLR, r16           ;set USB_SW_EN = '0', USB_SW_SEL = '0'

        ret
/*****************************************************************************
* Name:    OUT_CHAR
* Purpose:  Subroutine to output a single character to the transmit pin of the USART
* Inputs:   None
* Outputs:  Transmits data via USART
* Affected: USARTD0_STATUS, USARTD0_DATA
 *****************************************************************************/
OUT_CHAR:
        push r17

TX_POLL:
        lds r17, USARTD0_STATUS         ;load status register
        sbrs r17, 5                             ;proceed to writing out the char if
                                                                ;       the DREIF flag is set
        rjmp TX_POLL                            ;else go back to polling
        sts USARTD0_DATA, r16           ;send the character out over the USART

        pop r17

        ret
/*****************************************************************************
* Name:    OUT_STRING
* Purpose:  Subroutine to output character strings stored in memory
* Inputs:   None
* Outputs:  Transmits data via USART
* Affected: r16
 *****************************************************************************/
OUT_STRING:
        ldi ZL, low(STRING)
        ldi ZH, high(STRING)
PRINT_CHAR:
        ld r16, Z+                              ;load char pointed to by Z, POST-increment
        cpi r16, 0x00                           ;check if char is null
        breq PRINT_DONE                         ;       if null -> DONE printing string
        rcall OUT_CHAR                          ;       else OUTPUT that char
        rjmp PRINT_CHAR                         ;repeat

PRINT_DONE:
```

```
		ret
/*************************************************************************
* Name:    IN_CHAR
* Purpose:  Subroutine to receive a single character from receiver pin of the USART
* Inputs:   None
* Outputs:  r16 loaded with input from SCI
* Affected: r16, USARTD0_STATUS, USARTD0_DATA
 *************************************************************************/
IN_CHAR:

RX_POLL:
		lds r16, USARTD0_STATUS		;load the status register
		sbrs r16, 7				;proceed to reading in a char if
								;		the RXB8 flag is set
		rjmp RX_POLL				;else continue polling
		lds r16, USARTD0_DATA		;read the character into r16

		ret
```

Lab4_serial_baud_test.asm

```
/* Lab4_serial_baud_test.asm
 *
 * Lab 4 XMEGA USART Baud Rate Test
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to test the Baud Rate of the XMEGA's USART System.
 */

.nolist
.include "ATxmega128A1Udef.inc"
.list

.equ BSel = 289
.equ BScale = -7

.org 0x0000
		rjmp MAIN

.org 0x0200
MAIN:
		rcall GPIO_INIT
		rcall USART_INIT
		ldi r16, 0x55				;load U

REPEAT:
		rcall OUT_CHAR				;echo U to console
		rcall DELAY_1MS				;delay 1000us

		rjmp REPEAT
/*************************************************************************
* Name:    USART_INIT
```

```
* Purpose:  Subroutine to initialize the XMEGA USART system
* Inputs:   None
* Outputs:  None
* Affected: r16, USARTE0_CTRLB, USARTE0_CTRLC, USARTE0_BAUDCTRLA, USARTE0_BAUDCTRLB
 ***************************************************************************/
 USART_INIT:
            ldi r16, 0x18
            sts USARTE0_CTRLB, r16              ;turn on TXEN and RXEN lines

            ldi r16, 0x23
            sts USARTE0_CTRLC, r16              ;Parity = even, 8 bit frame, 1 stop bit

            ldi r16, (BSel & 0xFF)              ;select only lower 8 bits of BSel
            sts USARTE0_BAUDCTRLA, r16          ;set BAUDCTRLA to lower 8 bits of BSel

            ldi r16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)
            sts USARTE0_BAUDCTRLB, r16          ;set BAUDCTRLB to BScale | Bsel.
                                                ;           Lower 4 bits are upper 4 bits of BSel.
                                                ;           Upper 4 bits are upper 4 bits of BScale.
            ret
/***************************************************************************
* Name:    USART_INIT
* Purpose:  Subroutine to initialize the XMEGA USART system
* Inputs:   None
* Outputs:  None
* Affected: r16, PORTE_DIR, PORTE_OUT, PORTQ_DIR, PORTQ_OUT
 ***************************************************************************/
 GPIO_INIT:
            ldi r16, 0x08
            sts PORTE_DIRSET, r16               ;set PORTE_PIN3 as output for TX pin of USARTE0
            sts PORTE_OUTSET, r16               ;set the TX line to default to '1'

            ldi r16, 0x04
            sts PORTE_DIRCLR, r16               ;set RX pin for input

            ldi r16, 0xA
            sts PORTQ_DIRSET, r16               ;set pins 1 and 3 of PORTQ to output
            sts PORTQ_OUTCLR, r16               ;set USB_SW_EN = '0', USB_SW_SEL = '0'

            ret
/***************************************************************************
* Name:    OUT_CHAR
* Purpose:  Subroutine to output a single character to the transmit pin of the USART
* Inputs:   None
* Outputs:  Transmits data via USART
* Affected: USARTE0_STATUS, USARTE0_DATA
 ***************************************************************************/
OUT_CHAR:
            push r17

TX_POLL:
```

```
		lds r17, USARTE0_STATUS		;load status register
		sbrs r17, 5				;proceed to writing out the char if
								;		the DREIF flag is set
		rjmp TX_POLL				;else go back to polling
		sts USARTE0_DATA, r16		;send the character out over the USART

		pop r17

		ret
/*************************************************************************
* Name:    OUT_STRING
* Purpose:  Subroutine to output character strings stored in memory
* Inputs:   None
* Outputs:  Transmits data via USART
* Affected: r16
 *************************************************************************/
OUT_STRING:
PRINT_CHAR:
		ld r16, Z+				;load char pointed to by Z, POST-increment
		cpi r16, 0x00				;check if char is null
		breq PRINT_DONE				;		if null -> DONE printing string
		rcall OUT_CHAR				;		else OUTPUT that char
		rjmp PRINT_CHAR				;repeat

PRINT_DONE:
		ret
/*************************************************************************
* Name:    IN_CHAR
* Purpose:  Subroutine to receive a single character from receiver pin of the USART
* Inputs:   None
* Outputs:  r16 loaded with input from SCI
* Affected: r16, USARTE0_STATUS, USARTE0_DATA
 *************************************************************************/
IN_CHAR:

RX_POLL:
		lds r16, USARTE0_STATUS		;load the status register
		sbrs r16, 7				;proceed to reading in a char if
								;		the RXCIF flag is set
		rjmp RX_POLL				;else continue polling
		lds r16, USARTE0_DATA		;read the character into r16

		ret
/*************************************************************************
* Name:    DELAY_1MS
* Purpose:  Subroutine to delay for X*10ms
* Inputs:   X stored in r20
* Outputs:  None
* Affected: r20
 *************************************************************************/
```

```
.equ jcycles = 2
.equ kcycles = 246

DELAY_1MS:
        dec r20
        rcall JLOOP_INIT
        cpi r20, 0
        brne DELAY_1MS
        ret

JLOOP_INIT:
        push r20
        ldi r20, jcycles
JLOOP:
        dec r20
        rcall KLOOP_INIT
        cpi r20, 0
        brne JLOOP
        pop r20
        ret

KLOOP_INIT:
        push r20
        ldi r20, kcycles
KLOOP:
        dec r20
        nop
        brne KLOOP
        pop r20
        ret
```

Lab4_serial_menu.asm

```
/* Lab4_serial_menu.asm
 *
 * Lab 4 Serial Menu
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: The purpose of this program is to provide a menu with responses by using the XMEGA USART System.
 */

.nolist
.include "ATxmega128A1Udef.inc"
.list

.equ BSel = 289
.equ BScale = -7

.equ Prompt = '?'
.equ CR = 0x0D
.equ LF = 0x0A
.equ ESC = 0x1B
```

Lab: 4
Nicholas Imamshah
Section: 6957

```
.equ TAB = 0x09
.equ NUL = 0x00

.org 0x100
MENU:   .db "Nick's favorite:", CR, LF
                .db "1.", TAB, "OS/Computer (Mac or PC)", CR, LF
                .db "2.", TAB, "EE/CE Course ", CR, LF
                .db "3.", TAB, "Hobby", CR, LF
                .db "4.", TAB, "Quote", CR, LF
                .db "5.", TAB, "Movie", CR, LF
                .db "6.", TAB, "Re-display menu", CR, LF
                .db "ESC: exit", CR, LF, CR, LF, NUL
FAV_OS: .db     "Windows, unless Danny is looking, then Linux ", CR, LF, CR, LF, NUL
FAV_CE: .db     "Design Patterns", CR, LF, CR, LF, NUL
FAV_HO:.db "Gaming with friends", CR, LF, CR, LF, NUL
FAV_QT: .db "From an archery instructor: 'The reason you are missing, is because you are focusing on the target and not on your
actions.' ", CR, LF, CR, LF, NUL
FAV_MV:.db      "Kung Fury", CR, LF, CR, LF, NUL
PR_ESC:  .db    "Done!", CR, LF, CR, LF, NUL
.org 0x0000
        rjmp MAIN

.org 0x0200
MAIN:
        rcall GPIO_INIT
        rcall USART_INIT

REPEAT:
        ldi r16, Prompt                 ;load Prompt character
        ldi ZL, low(MENU << 1)
        ldi ZH, high(MENU << 1)
        rcall OUT_STRING

        rcall IN_CHAR                   ;read input
        rcall OUT_CHAR                  ;echo to console
        push r16                        ;push to stack
        ldi r16, CR                     ;CRLF
        rcall OUT_CHAR
        ldi r16, LF
        rcall OUT_CHAR
        pop r16                                 ;pop input off stack

        cpi r16, '1'
        breq OPT_1                              ;option 1 chosen

        cpi r16, '2'
        breq OPT_2                              ;option 2 chosen

        cpi r16, '3'
        breq OPT_3                              ;option 3 chosen
```

```
        cpi r16, '4'
        breq OPT_4                      ;option 4 chosen

        cpi r16, '5'
        breq OPT_5                      ;option 5 chosen

        cpi r16, '6'
        breq REPEAT                     ;option 6 chosen

        cpi r16, ESC
        breq OPT_E                      ;option ESC chosen

        rjmp REPEAT

OUTPUT:
        rcall OUT_STRING
        rjmp REPEAT

OPT_1:
        ldi ZL, low(FAV_OS << 1)
        ldi ZH, high(FAV_OS << 1)
        rjmp OUTPUT
OPT_2:
        ldi ZL, low(FAV_CE << 1)
        ldi ZH, high(FAV_CE << 1)
        rjmp OUTPUT
OPT_3:
        ldi ZL, low(FAV_HO << 1)
        ldi ZH, high(FAV_HO << 1)
        rjmp OUTPUT
OPT_4:
        ldi ZL, low(FAV_QT << 1)
        ldi ZH, high(FAV_QT << 1)
        rjmp OUTPUT
OPT_5:
        ldi ZL, low(FAV_MV << 1)
        ldi ZH, high(FAV_MV << 1)
        rjmp OUTPUT
OPT_E:
        ldi ZL, low(PR_ESC << 1)
        ldi ZH, high(PR_ESC << 1)
        rcall OUT_STRING
        rjmp DONE

DONE:
        rjmp DONE
/*****************************************************************************
* Name:    USART_INIT
* Purpose:  Subroutine to initialize the XMEGA USART system
* Inputs:   None
* Outputs:  None
```

```
* Affected: r16, USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA, USARTD0_BAUDCTRLB
 ***************************************************************************/
 USART_INIT:
        ldi r16, 0x18
        sts USARTD0_CTRLB, r16          ;turn on TXEN and RXEN lines

        ldi r16, 0x23
        sts USARTD0_CTRLC, r16          ;Parity = even, 8 bit frame, 1 stop bit

        ldi r16, (BSel & 0xFF)          ;select only lower 8 bits of BSel
        sts USARTD0_BAUDCTRLA, r16      ;set BAUDCTRLA to lower 8 bits of BSel

        ldi r16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)
        sts USARTD0_BAUDCTRLB, r16      ;set BAUDCTRLB to BScale | Bsel.
                                                ;       Lower 4 bits are upper 4 bits of BSel.
                                                ;       Upper 4 bits are upper 4 bits of BScale.
        ret
/***************************************************************************
* Name:    GPIO_INIT
* Purpose:  Subroutine to initialize the XMEGA GPIO for the USART System
* Inputs:  None
* Outputs:  None
* Affected: r16, PORTD_DIR, PORTD_OUT, PORTQ_DIR, PORTQ_OUT
 ***************************************************************************/
 GPIO_INIT:
        ldi r16, 0x08
        sts PORTD_DIRSET, r16           ;set PORTD_PIN3 as output for TX pin of USARTD0
        sts PORTD_OUTSET, r16           ;set the TX line to default to '1'

        ldi r16, 0x04
        sts PORTD_DIRCLR, r16           ;set RX pin for input

        ldi r16, 0xA
        sts PORTQ_DIRSET, r16           ;set pins 1 and 3 of PORTQ to output
        sts PORTQ_OUTCLR, r16           ;set USB_SW_EN = '0', USB_SW_SEL = '0'

        ret
/***************************************************************************
* Name:    OUT_CHAR
* Purpose:  Subroutine to output a single character to the transmit pin of the USART
* Inputs:  None
* Outputs:  Transmits data via USART
* Affected: USARTD0_STATUS, USARTD0_DATA
 ***************************************************************************/
OUT_CHAR:
        push r17

TX_POLL:
        lds r17, USARTD0_STATUS         ;load status register
        sbrs r17, 5                             ;proceed to writing out the char if
                                                ;       the DREIF flag is set
```

```
        rjmp TX_POLL                    ;else go back to polling
        sts USARTD0_DATA, r16           ;send the character out over the USART

        pop r17

        ret
/*************************************************************************
* Name:    OUT_STRING
* Purpose:  Subroutine to output character strings stored in memory
* Inputs:   Z pointing to desired output string
* Outputs:  Transmits data via USART
* Affected: r16
 *************************************************************************/
OUT_STRING:
        ;push r16
PRINT_CHAR:
        lpm r16, Z+                     ;load char pointed to by Z, POST-increment
        cpi r16, NUL                    ;check if char is null
        breq PRINT_DONE                 ;       if null -> DONE printing string
        rcall OUT_CHAR                  ;       else OUTPUT that char
        rjmp PRINT_CHAR                 ;repeat

        ;pop r16
PRINT_DONE:
        ret
/*************************************************************************
* Name:    IN_CHAR
* Purpose:  Subroutine to receive a single character from receiver pin of the USART
* Inputs:   None
* Outputs:  r16 loaded with input from SCI
* Affected: r16, USARTD0_STATUS, USARTD0_DATA
 *************************************************************************/
IN_CHAR:

RX_POLL:
        lds r16, USARTD0_STATUS         ;load the status register
        sbrs r16, 7                     ;proceed to reading in a char if
                                                ;       the RXCIF flag is set
        rjmp RX_POLL                    ;else continue polling
        lds r16, USARTD0_DATA           ;read the character into r16

        ret
```

Lab4_serial_int.asm

```
/* Lab4_serial_int.asm
 *
 * Lab 4 Serial Interrupt
 * Name: Nicholas Imamshah
 * Section: 6957
 * TA Name: Daniel Gonzalez
 * Description: This is an interrupt driven echo program.
 */
```

```
.nolist
.include "ATxmega128A1Udef.inc"
.list

.equ BSel = 289
.equ BScale = -7

.org 0x0000
		rjmp MAIN

.org USARTD0_RXC_VECT
		jmp ISR_ECHO

.org 0x0200
MAIN:
		rcall EBI_INIT
		rcall GPIO_INIT
		rcall USART_INIT
		rcall INTERRUPT_INIT

		ldi r16, 0x55
		st X, r16
		ldi r17, 0xFF					;for XORing to create a toggle

LOOP:
		EOR r16, r17					;toggle r16

		st X, r16					;output to LEDs

		ldi r20, 50
		rcall DELAY_X_MS

		rjmp LOOP
/*****************************************************************************
* Name:    OUT_CHAR
* Purpose:  Subroutine to output a single character to the transmit pin of the USART
* Inputs:   None
* Outputs:  Transmits data via USART
* Affected: USARTD0_STATUS, USARTD0_DATA
 *****************************************************************************/
OUT_CHAR:
		push r17

TX_POLL:
		lds r17, USARTD0_STATUS		;load status register
		sbrs r17, 5					;proceed to writing out the char if
									;		the DREIF flag is set
		rjmp TX_POLL				;else go back to polling
		sts USARTD0_DATA, r16		;send the character out over the USART
```

```
		pop r17

		ret
/*************************************************************************
* Name:    USART_INIT
* Purpose:  Subroutine to initialize the XMEGA USART system
* Inputs:   None
* Outputs:  None
* Affected: r16, USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA, USARTD0_BAUDCTRLB
 *************************************************************************/
 USART_INIT:
		ldi r16, 0x10
		sts USARTD0_CTRLA, r16		;enable low-level interrupts on USART receive

		ldi r16, 0x18
		sts USARTD0_CTRLB, r16		;turn on TXEN and RXEN lines

		ldi r16, 0x23
		sts USARTD0_CTRLC, r16		;Parity = even, 8 bit frame, 1 stop bit

		ldi r16, (BSel & 0xFF)		;select only lower 8 bits of BSel
		sts USARTD0_BAUDCTRLA, r16	;set BAUDCTRLA to lower 8 bits of BSel

		ldi r16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)
		sts USARTD0_BAUDCTRLB, r16	;set BAUDCTRLB to BScale | Bsel.
									;		Lower 4 bits are upper 4 bits of BSel.
									;		Upper 4 bits are upper 4 bits of BScale.
		ret
/*************************************************************************
* Name:    GPIO_INIT
* Purpose:  Subroutine to initialize the XMEGA GPIO for the USART System
* Inputs:   None
* Outputs:  None
* Affected: r16, PORTD_DIR, PORTD_OUT, PORTQ_DIR, PORTQ_OUT
 *************************************************************************/
 GPIO_INIT:
		ldi r16, 0x08
		sts PORTD_DIRSET, r16		;set PORTD_PIN3 as output for TX pin of USARTD0
		sts PORTD_OUTSET, r16		;set the TX line to default to '1'

		ldi r16, 0x04
		sts PORTD_DIRCLR, r16		;set RX pin for input

		ldi r16, 0xA
		sts PORTQ_DIRSET, r16		;set pins 1 and 3 of PORTQ to output
		sts PORTQ_OUTCLR, r16		;set USB_SW_EN = '0', USB_SW_SEL = '0'

		ret
/*************************************************************************
* Name:    EBI_INIT
* Purpose:  Subroutine to initialize Ports H,J,K/EBI for additional Input/Output ports
```

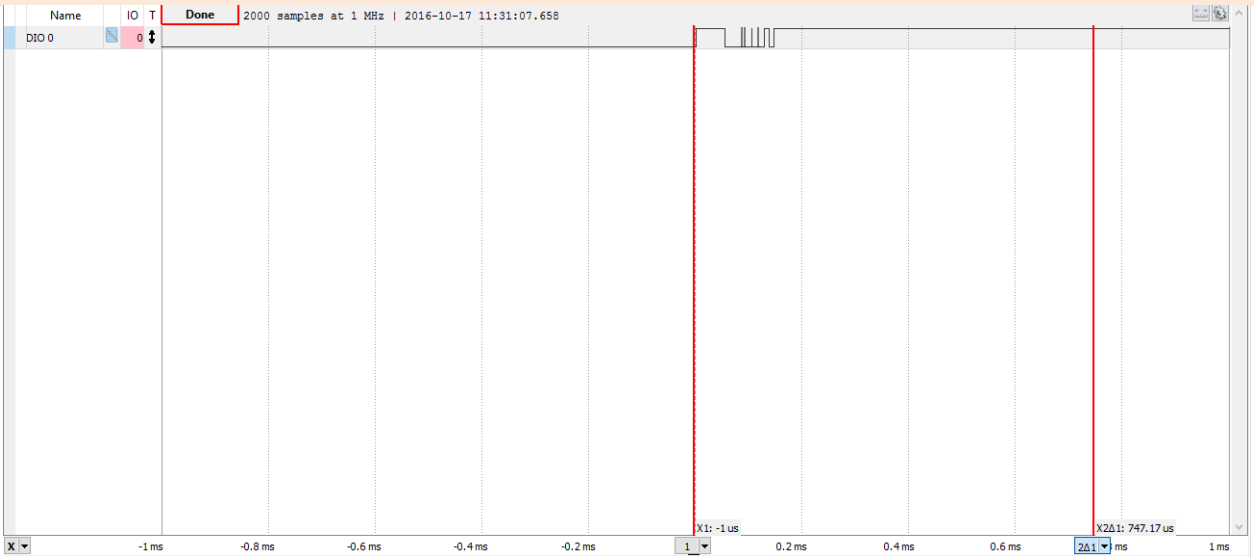
```
* Inputs:  None
* Outputs:  None
* Affected: R16, X, Z, EBI_CTRL, PORTH,J,K,
 ********************************************************************************/
.set PORT = 0x288000
.set PORT_END = 0x289FFF

EBI_INIT:
        ldi r16, 0x17                   ;Configure PORTH bits 4, 2, 1, and 0 as outputs.
        sts PORTH_DIRSET, r16   ;       These are the CS0(L), ALE1(H), RE(L), and WE(L) outputs.
                                                ;       (CS0 is bit 4; ALE1 is bit 2; RE is bit 1; WE is bit 0)

        ldi r16, 0x13                   ;Default CS0(L), RE(L), and WE(L) to H = false.
        sts PORTH_OUTSET, r16   ;       ALE defaults to 0 = L = false.

        ldi r16, 0xFF                   ;Set all PORTK pins (A15-A0) to be outputs.
        sts PORTK_DIRSET, r16

        ldi r16, 0xFF                   ;Set all PORTJ pins (D7-D0) to be outputs.
        sts PORTJ_DIRSET, r16

        ldi r16, 0x01                   ;Store 0x41 in EBI_CTRL reg to select 3 port EBI(H,J,K)
        sts EBI_CTRL, r16       ;       mode and SRAM ALE1 mode

;Initialize the Z pointer to point to the base address for CS0 in memory
        ldi ZH, high(EBI_CS0_BASEADDR)
        ldi ZL, low(EBI_CS0_BASEADDR)

;Load the middle byte (A15:8) of the three byte addr into a reg and store it as the
;       LOW byte of the Base Address, BASEADDRL.
        ldi r16, byte2(PORT)
        st Z+, r16

;Load the highest byte (A23:16) of the three byte addr into a reg and store it as the
;       HIGH byte of the Base Address, BASEADDRH.
        ldi r16, byte3(PORT)
        st Z, r16

        ldi r16, 0x15                   ;Set to 8KB CS space and turn on SRAM mode, 0x288000 - 0x289FFF
        sts EBI_CS0_CTRLA, r16

;Steps for using the port expansion
        ldi r16, byte3(PORT)    ;initialize a pointer to point to the base addr of the PORT
        sts CPU_RAMPX, r16              ;use the CPU_RAMPX reg to set the third byte of the pointer

        ldi XH, high(PORT)      ;set the middle (XH) and low (XL) bytes of the pointer as usual
        ldi XL, low(PORT)

        ret
/********************************************************************************
* Name:    INTERRUPT_INIT
```

```
* Purpose:  Subroutine to initialize the PortE external pin interrupt PE0 using INT0
* Inputs:   None
* Outputs:  None
* Affected: r16, PMIC_CTRL, PORTE: _INT0MASK_OUT, _DIRCLR, _INTCTRL, _PIN0CTRL
 ***************************************************************************/
INTERRUPT_INIT:
        ldi r16, 0x01                   ;enable low-level interrupts
        sts PMIC_CTRL, r16

        sei                                             ;set global interrupt flag LAST!
        ret
/***************************************************************************
* Name:     ISR_ECHO
* Purpose:  Interrupt service routine to echo char on USART receive pin to transmit
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_STATUS
 ***************************************************************************/
ISR_ECHO:
        push r16                        ;push any registers used to ensure they can be used after

        lds r16, USARTD0_DATA       ;read character into r16
        rcall OUT_CHAR                  ;echo character

        ldi r16, 0x80
        sts USARTD0_STATUS, r16     ;ensure RXCIF is cleared

        pop r16                                 ;restore registers used in ISR
        reti
/***************************************************************************
* Name:     DELAY_X_MS
* Purpose:  Subroutine to delay for X*10ms
* Inputs:   X stored in r20
* Outputs:  None
* Affected: r20
 ***************************************************************************/
.equ jcycles = 20
.equ kcycles = 246

DELAY_X_MS:
        dec r20
        rcall JLOOP_INIT
        cpi r20, 0
        brne DELAY_X_MS
        ret

JLOOP_INIT:
        push r20
        ldi r20, jcycles
JLOOP:
        dec r20
```

```
        rcall KLOOP_INIT
        cpi r20, 0
        brne JLOOP
        pop r20
        ret

KLOOP_INIT:
        push r20
        ldi r20, kcycles
KLOOP:
        dec r20
        nop
        brne KLOOP
        pop r20
        ret
```

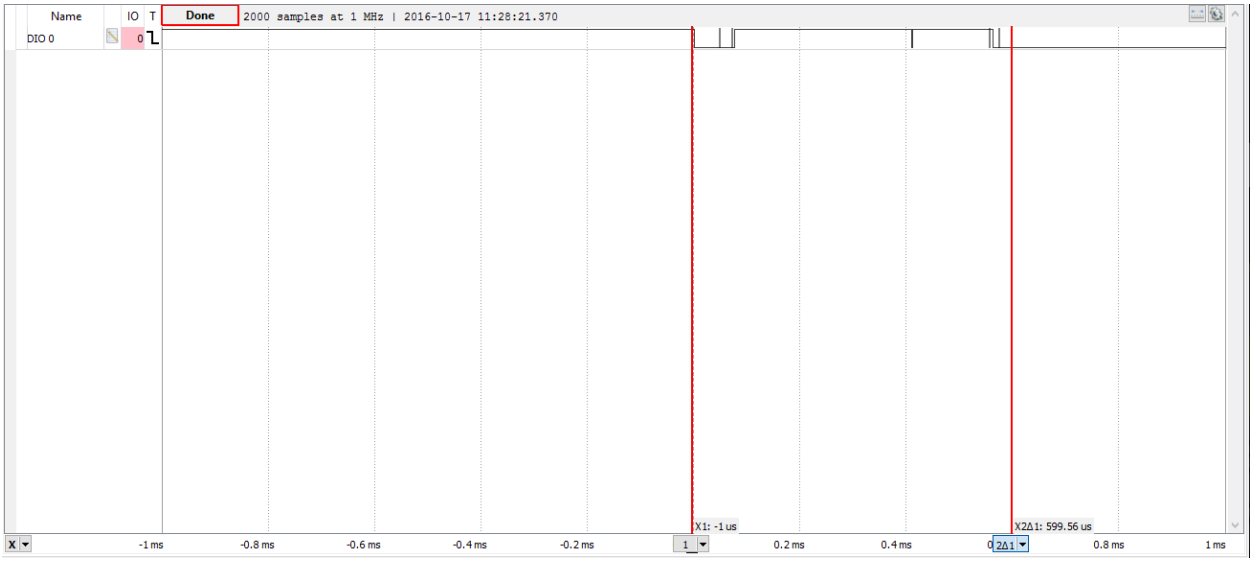
i) Appendix

Low-to-High Bouncing



High-to-Low Bouncing

Lab: 4
Nicholas Imamshah
Section: 6957



Baud Rate Verification