Lab 4

Michael Arboleda
Lab Section: 7F34

July 5, 2017

b. Answers to all pre-lab questions

Part A Question: Determine which pins on PORTD are used for USARTD0?
Ans: Pin 2 and Pin 3 are used for USARTD0

1) What is the difference between serial and parallel communication?

ANS: Serial using one pin, while parallel uses multiple pins

2) What is the difference between synchronous and asynchronous communication?

ANS: Synchronous communication means the transmitter and receiver are on the same clock. Asynchronous communication use Start and Stop bits and Baud rate to determine the speed and also uses parity bit for error checking

3) List the XMEGAs USART registers used in your programs and briefly describe their functions.

ANS:
DATA: This contains the data that was transmitted/received
STATUS: Contains flags like Receive Complete Interrupt Flag, Transfer Complete Interrupt Flag, and Data Register Empty Interrupt Flag.
CTRLC: Used to set up serial mode, width of data, parity, and number of stop bits
CTRLB: Used to enable properties of USARTD0 such as receive and transmit
CTRLA: Used to set up the level of the different types of interrupts for USARTD0
BAUDCTRLB: Sets the BSCALE in the 4 upper bits [7:4] and sets bit 11:8 for BSEL in 4 lower bits [3:0]
BAUDCTRLA: Sets bit 7:0 for BSEL in the respective order

4) What is the maximum possible baud you can use for asynchronous communication if your board runs at 32 MHz? Support your answer with the values you would place in any special registers that are needed.

ANS: Setting BAUD control A and B (BAUDCTRLA and BAUDCTRLB) to 0b00000000 would make the Baud rate 2Mb per second

c. Problems Encountered

My logic for ignoring invalid chars in part D was incorrect. I forgot to pop registers I had pushed.

d. Future Work/Applications

In this lab only one configuration for serial. It would be useful to try and make working programs for different serial configuration. I could also combine switches so i could set up the ascii value on the switches and transmit the corresponding character.

e. Schematics

N/A

g. Pseudocode/Flowcharts

Pseudocode for lab4_serial.asm:

```
MAIN:
    * Equate numbers
    * Set registers to hold constants
    * Call Change_CLK_32HZ subroutine

WHILE(TRUE){}
END



SUBROUTINE USART_INIT
    * Set Pins to Transmit and Recieve
    * Set up USART controls
    * Set up Baud Rate controls
    * Return to program

SUBROUTINE OUT_CHAR
    While(Transmitting){}
    * Transmit USART Data in R1
    * Return to program

SUBROUTINE OUT_STRING
    While(Z-point does not equal NULL){
        * increment Z-pointer
        * Store Z-pointer Data in R1
     * Call OUT_CHAR
    }
    * Return to program

SUBROUTINE IN_CHAR
    While(Transmitting){}
    * Store USART Data into R1
    * Return to program

SUBROUTINE Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
    * Return to program
```

Pseudocode for lab4_serial_baud_test.asm:

```
MAIN:
    * Equate numbers
    * Set registers to hold constants
    * Call Change_CLK_32HZ subroutine
    * LOAD 'U' into R1
WHILE(TRUE){}
    * Call OUT_CHAR
END

SUBROUTINE USART_INIT
    * Set Pins to Transmit and Recieve
    * Set up USART controls
    * Set up Baud Rate controls
    * Return to program

SUBROUTINE OUT_CHAR
    While(Transmitting){}
    * Transmit USART Data in R1
    * Return to program
```

Pseudocode for lab4_serial_menu.asm:

```
MAIN:
    * Equate numbers
    * Set registers to hold constants
    * Call Change_CLK_32HZ subroutine
    * Call Display_Menu
WHILE(TRUE){}
    * Call IN_CHAR
    * Call Z_POINTER_LOGIC
    if(Valid Choice){
        * Display Menu
    }
END

SUBROUTINE Z_POINTER_LOGIC
* Set Valid choice as true
    IF(Data = 1){
        * Point Z-Pointer to Food string
    }
    ELSE IF(Data = 2){
        * Point Z-Pointer to Quote string
    }
    ELSE IF(Data = 3){
        * Point Z-Pointer to Movie string
    }
    ELSE IF(Data = 4){
        * Point Z-Pointer to UF Course string
    }
    ELSE IF(Data = 5){
        * Point Z-Pointer to Hobby string
    }
    ELSE IF(Data = 6){
        * Return to program
    }
    ELSE IF(Data = D or Data = d){
     WHILE(TRUE){}
    }
    ELSE{
        * Set Valid choice as false
        * Return to program
    }
    * Call OUT_STRING
    * Return to program
```

```
SUBROUTINE Display_Menu
     * Point Z-Pointer to Menu string
     * Call OUT_STRING
     * Return to program


SUBROUTINE USART_INIT
    * Set Pins to Transmit and Recieve
    * Set up USART controls
    * Set up Baud Rate controls
    * Return to program


SUBROUTINE OUT_CHAR
    While(Transmitting){}
    * Transmit USART Data in R1
    * Return to program


SUBROUTINE OUT_STRING
    While(Z-point does not equal NULL){
        * increment Z-pointer
        * Store Z-pointer Data in R1
        * Call OUT_CHAR
    }
    * Return to program


SUBROUTINE IN_CHAR
    While(Transmitting){}
    * Store USART Data into R1
    * Return to program


SUBROUTINE Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
    * Return to program
```

Pseudocode for lab4_serial_int.asm:

```
MAIN:
    * Equate numbers
    * Set registers to hold constants
    * Call Change_CLK_32HZ subroutine
* Call Counter_INIT subroutine
* Call USART_INIT subroutine

WHILE(TRUE){}
if(Counter = 0){
    * Toggle LED ON/OFF
}
END

SUBROUTINE Counter_INIT
    * Set TOP(PER) of counter
    * Set Clock Prescalar
    * Return to program

ISR USART_ISR
    * STORE USART DATA in R1
    * Clear interrupt flag
    * Call OUT_CHAR
    * Return

SUBROUTINE USART_INIT
    * Set Pins to Transmit and Recieve
    * Set up USART controls
    * Set up Baud Rate controls
    * Return to program

SUBROUTINE OUT_CHAR
    While(Transmitting){}
    * Transmit USART Data in R1
    * Return to program

SUBROUTINE Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
    * Return to program
```

h. Program Code

Code for lab4_serial.asm:

```
; Lab 4 Part B
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Revices/Transmits Chars
;
; lab4_serial.asm

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ TxRx_On = 0b00011000        ; 0x18
.equ pin_Tx  = 0b00001000        ; 0x08
.equ pin_Rx  = 0b00000100        ; 0x04
; asynch, 8 databits, odd parity, 1 start, and 1 stop
.equ usart_ctrl_C        = 0b00110011
; BSEL 576000
.equ upper_BSEL          = 0b00000100
.equ BSEL                          = 0b00110111
.equ BSCALE                        = 0b10110000     ; -5  BSCALE
.equ NULL                          = 0x00           ; Null character

; Reg Defs
.def char_out = R1

;ORG defs
.ORG 0x0000                        ;Code starts running from address 0x0000.
        rjmp MAIN                  ;Relative jump to start of program.


.org 0x0200

Test_String:
.db "Test for OUT_STRING", NULL

MAIN:

        call Change_CLK_32HZ            ; Change Clock Speed
        call USART_INIT                 ; Call to set up USART
; START TEST
```

```
        ldi ZL, low(Test_String << 1) ; Set Z-Pointer to Test String
        ldi ZH, high(Test_String << 1) ; Set Z-Pointer to Test String
        call OUT_STRING                 ; Call out string to send string
; END TEST

; infinite loop
Never_End:

; START TEST
        call IN_CHAR    ; Recieve Char
        call OUT_CHAR   ; Transmit Char
; END TEST
        rjmp Never_End          ; Jump to restart output loop





;*******************SUBROUTINES************************************
; Subroutine Name:  USART_INIT
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
USART_INIT:

        push R16                                ; Push R16

        ldi R16, pin_Tx                 ; Pin3 config
        sts PortD_DIRSET, R16   ; Set Pin3 to transmit
        sts PortD_OUTSET, R16   ; Output High to Pin3


        ldi R16, pin_Rx                 ; Pin2 config
        sts PORTD_DIRCLR, R16   ; Set Pin2 to recieve

        ; Set Control B
        ldi R16, TxRx_On                ; LOAD Tx and Rx config
        sts USARTD0_CTRLB, R16          ; Set Tx, Rx lines

        ; Set Control C
        ldi R16, usart_ctrl_C           ; LOAD CTRL config
        sts USARTD0_CTRLC, R16          ; Set USART

        ; Set Baud Rate Ctrl B
        ldi R16, (BSCALE | upper_BSEL); OR BSCALE with 11:8 bit of BSEL
        sts USARTD0_BAUDCTRLB, R16 ; Set BAUD CTRL B
        ; Set Baud Rate Ctrl A
        ldi R16, BSEL                 ; LOAD 7:0 of BSEL in  R16
        sts USARTD0_BAUDCTRLA, R16            ; Set BAUD CTRL A

        pop R16                               ; POP r16
        ret                                   ; Return



;********************SUBROUTINES*************************************
; Subroutine Name:  OUT_CHAR
; Inputs: R1
; Outputs: No direct outputs
; Affected: None

OUT_CHAR:
        push R16                                      ; Push R16

Transfer_Complete:
        lds R16, USARTD0_STATUS       ; LOAD status reg to R16
        sbrs R16, 5            ; Skip jump if bit 5 is 1
        rjmp Transfer_Complete        ; Restart Loop
        sts USARTD0_DATA, R1          ; Output char thru USART

        pop R16                       ; POP R16
        ret                                   ; Return

;********************SUBROUTINES*************************************
; Subroutine Name: OUT_STRING
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
OUT_STRING:
        push R16        ; Push R16

While_String:
        lpm R16, Z+            ; LOAD Z data and increment pointer
        cpi R16, NULL          ; Compare Data and Null char
        breq End_While_String  ; If data = null, branch to exit
        mov R1, R16                   ; move R16 Data to R1
        call OUT_CHAR                 ; Call OUT_CHAR
        jmp While_String              ; Restart Loop

End_While_String:
        pop r16          ; Pop R16
        ret              ; Return


;*******************SUBROUTINES*************************************
; Subroutine Name:  IN_CHAR
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
IN_CHAR:

        push R16                                              ; PUSH R16

Recieve_Complete:
        lds  R16, USARTD0_STATUS        ; LOAD the status register
        sbrs R16, 7             ; Skip jump if bit 7 is 1
        rjmp Recieve_Complete           ; Restart Loop
        lds  R1, USARTD0_DATA           ; LOAD data into R1

        pop R16          ; POP r16
        ret              ; Return


;*******************SUBROUTINES*************************************
; Subroutine Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Change_CLK_32HZ:
        ;Push Values
        push R17 ; PUSH r17 to stack
        push R18 ; PUSH r18 to stack
        push R19 ; PUSH r19 to stack
        push R20 ; PUSH r20 to stack

; Enable the new oscillator
ldi R16, new_clock_freq         ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16                       ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg
; While flag is not set
While_32_flag:
        lds R17, OSC_STATUS             ; Load Status Flag
        and R17, R16                    ; Bit-mask with 00000010
        cp R17, R16                     ; Compare Mask and Value
        brne While_32_flag              ; Restart loop if flag not set
```

```
; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc    ; Load IOREG into R17
        sts CPU_CCP, R17                  ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17                 ; Store config in clk control

        ;Pop Values
        pop R20 ; POP r20 from stack
        pop R19 ; POP r19 from stack
        pop R18 ; POP r18 from stack
        pop R17 ; POP r17 from stack
        ret
```

Code for lab4_serial_baud_test.asm:

```
; Lab 4 Part C
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Revices/Transmits Chars
;
; lab4_serial_baud_test.asm
; Created: 7/5/2017 5:18:11 AM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ TxRx_On = 0b00011000        ; 0x18
.equ pin_Tx  = 0b00001000        ; 0x08
.equ pin_Rx  = 0b00000100        ; 0x04
; asynch, 8 databits, odd parity, 1 start, and 1 stop
.equ usart_ctrl_C       = 0b00110011
; BSEL 576000
.equ upper_BSEL         = 0b00000100
.equ BSEL                         = 0b00110111
.equ BSCALE                       = 0b10110000     ; -5  BSCALE
.equ NULL                         = 0x00           ; Null character

; Reg Defs
.def char_out = R1

;ORG defs
.ORG 0x0000                        ;Code starts running from address 0x0000.
        rjmp MAIN                  ;Relative jump to start of program.


.org 0x0200

Test_String:
.db "Test_for_OUT_STRING", NULL

MAIN:

        call Change_CLK_32HZ     ; Change Clk speed
        call USART_INIT          ; Initilize USART
        ldi r16, 'U'                        ; LOAD 'U' data into R16
        mov R1, R16                         ; MOVE 'U' to R1
; infinite loop
Never_End:

        call OUT_CHAR           ; Output 'U'

        rjmp Never_End          ; Jump to restart output loop


;*******************SUBROUTINES************************************
; Subroutine Name:  USART_INIT
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
USART_INIT:

        push R16                                    ; Push R16

        ldi R16, pin_Tx                     ; Pin3 config
        sts PortC_DIRSET, R16   ; Set Pin3 to transmit
        sts PortC_OUTSET, R16   ; Output High to Pin3


        ldi R16, pin_Rx                     ; Pin2 config
        sts PORTC_DIRCLR, R16   ; Set Pin2 to recieve

        ; Set Control B
        ldi R16, TxRx_On                    ; LOAD Tx and Rx config
        sts USARTC0_CTRLB, R16              ; Set Tx, Rx lines

        ; Set Control C
        ldi R16, usart_ctrl_C               ; LOAD CTRL config
        sts USARTC0_CTRLC, R16              ; Set USART

        ; Set Baud Rate Ctrl B
        ldi R16, (BSCALE | upper_BSEL) ; OR BSCALE with 11:8 bit of BSEL
        sts USARTC0_BAUDCTRLB, R16                  ; Set BAUD CTRL B

        ; Set Baud Rate Ctrl A
        ldi R16, BSEL                       ; LOAD 7:0 of BSEL in  R16
        sts USARTC0_BAUDCTRLA, R16                  ; Set BAUD CTRL A

        pop R16                 ; POP r16
        ret                                 ; Return
```

```
;*******************SUBROUTINES*********************************
; Subroutine Name:  OUT_CHAR
; Inputs: R1
; Outputs: No direct outputs
; Affected: None

OUT_CHAR:
        push R16                                        ; Push R16

Transfer_Complete:
        lds R16, USARTC0_STATUS         ; LOAD status reg to R16
        sbrs R16, 5                     ; Skip jump if bit 5 is 1
        rjmp Transfer_Complete          ; Restart Loop
        sts USARTC0_DATA, R1            ; Output char thru USART

        pop R16         ; POP R16
        ret                     ; Return

;*******************SUBROUTINES*********************************
; Subroutine Name: OUT_STRING
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None

OUT_STRING:
        push R16        ; Push R16

While_String:
        lpm R16, Z+             ; LOAD Z data and increment pointer
        cpi R16, NULL   ; Compare Data and Null char
        breq End_While_String   ; If data = null, branch to exit
        mov R1, R16                     ; move R16 Data to R1
        call OUT_CHAR           ; Call OUT_CHAR
        jmp While_String        ; Restart Loop

End_While_String:

        pop r16         ; Pop R16
        ret                     ; Return


;*******************SUBROUTINES*********************************
; Subroutine Name:  IN_CHAR
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None

IN_CHAR:

        push R16                                                ; PUSH R16

Recieve_Complete:
        lds  R16, USARTD0_STATUS          ; LOAD the status register
        sbrs R16, 7                       ; Skip jump if bit 7 is 1
        rjmp Recieve_Complete             ; Restart Loop
        lds  R1, USARTD0_DATA             ; LOAD data into R1

        pop R16                   ; POP r16
        ret                               ; Return


;********************SUBROUTINES************************************
; Subroutine Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Change_CLK_32HZ:
        ;Push Values
        push R17 ; PUSH r17 to stack
        push R18 ; PUSH r18 to stack
        push R19 ; PUSH r19 to stack
        push R20 ; PUSH r20 to stack

; Enable the new oscillator
ldi R16, new_clock_freq           ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16                         ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg
; While flag is not set
While_32_flag:
        lds R17, OSC_STATUS               ; Load Status Flag
        and R17, R16                      ; Bit-mask with 00000010
        cp R17, R16                       ; Compare Mask and Value
        brne While_32_flag                ; Restart loop if flag not set

; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc     ; Load IOREG into R17
        sts CPU_CCP, R17                  ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
```

```
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17              ; Store config in clk control

        ;Pop Values
        pop R20 ; POP r20 from stack
        pop R19 ; POP r19 from stack
        pop R18 ; POP r18 from stack
        pop R17 ; POP r17 from stack
        ret
```

Code for lab4_serial_menu.asm:

```
; Lab 4 Part D
; Name:          Michael Arboleda
; Section:       7F34
; TA Name:       Wesley Piard
; Description: Revices/Transmit data for menu

; lab4_serial_menu.asm
; Created: 7/4/2017 6:15:54 PM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ TxRx_On = 0b00011000        ; 0x18
.equ pin_Tx  = 0b00001000        ; 0x08
.equ pin_Rx  = 0b00000100        ; 0x04
; asynch, 8 databits, odd parity, 1 start, and 1 stop
.equ usart_ctrl_C       = 0b00110011
; BSEL 576000
.equ upper_BSEL         = 0b00000100
.equ BSEL                       = 0b00110111
.equ BSCALE                     = 0b10110000     ; -5  BSCALE
.equ NULL                       = 0x00           ; Null character
.equ CR                         = 0x0D           ; Carriage Return
.equ LF                         = 0x0A           ; Line Feed

; Reg Defs
.def char_out = R1

;ORG defs
.ORG 0x0000                      ;Code starts running from address 0x0000.
        rjmp MAIN                ;Relative jump to start of program.


.org 0x0200

Test_String:
.db "Test for OUT_STRING", NULL

Menu:
.db "Michael Arboleda's favorite:", LF, CR, 0x09, "1. Food", LF, CR,\
0x09, "2. Quote", LF, CR, 0x09, "3. Movie", LF, CR,\
0x09, "4. UF Course", LF, CR, 0x09, "5. Hobby", LF, CR,\
0x09, "6. Re-display menu", LF, CR, 0x09, "D: Done", LF, CR, NULL

Food:
.db "Michael Arboleda's favorite Food is steak", LF, CR, NULL

Quote:
.db "Michael Arboleda's favorite Quote is ", 0x22, "I know nothing!",\
0x22, " - Michael Scott", LF, CR, NULL

Movie:
.db "Michael Arboleda's favorite Movie is Scott Pilgrim Vs. The World",\
LF, CR, NULL

UF_Course:
.db "Michael Arboleda's favorite UF Course is MAA4212, Advanced Calculus 2",\
LF, CR, NULL

Hobby:
.db "Michael Arboleda's favorite Hobby is playing Xbox", LF, CR, NULL


MAIN:

        call Change_CLK_32HZ     ; Change Clk to 32MHz
        call USART_INIT                  ; Initilize USART

        ldi R20, 0x00                    ; LOAD R20 with 0

        call Display_Menu                ; Transmit menu


; infinite loop
Never_End:
        call IN_CHAR                     ; Receive Char
        call Z_POINTER_LOGIC     ; Set Z-pointer

        sbrs R20, 0                              ; Skip is R20 is 0
        call Display_Menu                ; Transmit menu

        rjmp Never_End           ; Jump to restart output loop



;*******************SUBROUTINES***********************************
; Subroutine Name:  Z_POINTER_LOGIC
; Inputs: No direct input (from stack)
; Outputs: R20, Z-Pointer
; Affected: None
Z_POINTER_LOGIC:
        push R16 ; PUSH R16
        push R17 ; PUSH R17

        mov R16, R1             ; Move R1 into R16

        ldi R20, 0x00   ; Set R20 to 0
; IF '1'
        cpi R16, '1'                    ; Compare Char to 1
        brne IF2
        ldi ZL, low(Food << 1)  ; Set Lower bits for Z-pointer
        ldi ZH, high(Food << 1) ; Set upper bits for Z-pointer
        jmp END_IF                              ; JUMP to end switch
; Else if '2'
IF2:
        cpi R16, '2'                            ; Compare Char to 2
        brne IF3
        ldi ZL, low(Quote << 1)         ; Set Lower bits for Z-pointer
        ldi ZH, high(Quote << 1)        ; Set upper bits for Z-pointer
        jmp END_IF                              ; JUMP to end switch
; Else if '3'
IF3:
        cpi R16, '3'                            ; Compare Char to 3
        brne IF4
        ldi ZL, low(Movie << 1)         ; Set Lower bits for Z-pointer
        ldi ZH, high(Movie << 1)        ; Set upper bits for Z-pointer
        jmp END_IF                                      ; JUMP to end switch
; Else if '4'
IF4:
        cpi R16, '4'                            ; Compare Char to 4
        brne IF5
        ldi ZL, low(UF_Course << 1)     ; Set Lower bits for Z-pointer
        ldi ZH, high(UF_Course << 1)    ; Set upper bits for Z-pointer
        jmp END_IF              ; JUMP to end switch
; Else if '5'
IF5:
        cpi R16, '5'                    ; Compare Char to 5
        brne IF6
        ldi ZL, low(Hobby << 1) ; Set Lower bits for Z-pointer
        ldi ZH, high(Hobby << 1)        ; Set upper bits for Z-pointer
        jmp END_IF                              ; JUMP to end switch
; Else if '6'
IF6:
        cpi R16, '6'                        ; Compare Char to 6
        brne IFD
        pop r17                                     ; POP R17
        pop r16                                     ; POP R16
        ret                                                 ; return
;Else if 'D' or 'd'
IFD:
        cpi R16, 'D'                        ; Compare Char to D
        breq PASS                                   ; Start infinite loop
        cpi R16, 'd'                        ; Compare Char to d
        brne ELSE                                   ; JUMP to else
PASS:
        jmp PASS

; else
ELSE:
        ldi R20, 0x01                       ; Set R20 to 1
        pop r17                                     ; POP R17
        pop r16                                     ; POP R16
        ret                                                 ; return

END_IF:
        call OUT_STRING                     ; Call out_string

        pop R17 ; POP R17
        pop R16 ; POP R16
        ret             ; Return



;*******************SUBROUTINES***********************************
; Subroutine Name:  Display_Menu
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Display_Menu:
        ; Display Menu
        ldi ZL, low(Menu << 1)   ; Set Lower bits for Z-pointer
        ldi ZH, high(Menu << 1)  ; Set upper bits for Z-pointer
        call OUT_STRING                     ; Output String
        ret                                 ; Return
```

```
;*******************SUBROUTINES*************************************
; Subroutine Name:  USART_INIT
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
USART_INIT:

        push R16                                ; Push R16

        ldi R16, pin_Tx                 ; Pin3 config
        sts PortD_DIRSET, R16   ; Set Pin3 to transmit
        sts PortD_OUTSET, R16   ; Output High to Pin3


        ldi R16, pin_Rx                 ; Pin2 config
        sts PORTD_DIRCLR, R16   ; Set Pin2 to recieve

        ; Set Control B
        ldi R16, TxRx_On                        ; LOAD Tx and Rx config
        sts USARTD0_CTRLB, R16          ; Set Tx, Rx lines

        ; Set Control C
        ldi R16, usart_ctrl_C           ; LOAD CTRL config
        sts USARTD0_CTRLC, R16          ; Set USART

        ; Set Baud Rate Ctrl B
        ldi R16, (BSCALE | upper_BSEL) ; OR BSCALE with 11:8 bit of BSEL
        sts USARTD0_BAUDCTRLB, R16              ; Set BAUD CTRL B

        ; Set Baud Rate Ctrl A
        ldi R16, BSEL                   ; LOAD 7:0 of BSEL in  R16
        sts USARTD0_BAUDCTRLA, R16              ; Set BAUD CTRL A

        pop R16                         ; POP r16
        ret                                     ; Return



;*******************SUBROUTINES*************************************
; Subroutine Name:  OUT_CHAR
; Inputs: R1
; Outputs: No direct outputs
; Affected: None
```

```
OUT_CHAR:
        push R16                                        ; Push R16

Transfer_Complete:
        lds R16, USARTD0_STATUS         ; LOAD status reg to R16
        sbrs R16, 5                     ; Skip jump if bit 5 is 1
        rjmp Transfer_Complete          ; Restart Loop
        sts USARTD0_DATA, R1            ; Output char thru USART

        pop R16                 ; POP R16
        ret                             ; Return

;*******************SUBROUTINES***********************************
; Subroutine Name: OUT_STRING
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
OUT_STRING:
        push R16          ; Push R16

While_String:
        lpm R16, Z+             ; LOAD Z data and increment pointer
        cpi R16, NULL                   ; Compare Data and Null char
        breq End_While_String   ; If data = null, branch to exit
        mov R1, R16             ; move R16 Data to R1
        call OUT_CHAR                   ; Call OUT_CHAR
        jmp While_String                ; Restart Loop

End_While_String:

        pop r16           ; Pop R16
        ret               ; Return


;*******************SUBROUTINES***********************************
; Subroutine Name:  IN_CHAR
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
IN_CHAR:

        push R16                                        ; PUSH R16

Recieve_Complete:
        lds  R16, USARTD0_STATUS        ; LOAD the status register
        sbrs R16, 7             ; Skip jump if bit 7 is 1
        rjmp Recieve_Complete              ; Restart Loop
        lds  R1, USARTD0_DATA              ; LOAD data into R1

        pop R16          ; POP r16
        ret              ; Return

;*******************SUBROUTINES**************************************
; Subroutine Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Change_CLK_32HZ:
        ;Push Values
        push R17 ; PUSH r17 to stack
        push R18 ; PUSH r18 to stack
        push R19 ; PUSH r19 to stack
        push R20 ; PUSH r20 to stack

; Enable the new oscillator
ldi R16, new_clock_freq        ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16                          ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg
; While flag is not set
While_32_flag:
        lds R17, OSC_STATUS                ; Load Status Flag
        and R17, R16                       ; Bit-mask with 00000010
        cp R17, R16                        ; Compare Mask and Value
        brne While_32_flag                 ; Restart loop if flag not set

; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc    ; Load IOREG into R17
        sts CPU_CCP, R17                   ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17                  ; Store config in clk control

        ;Pop Values
        pop R20 ; POP r20 from stack
        pop R19 ; POP r19 from stack
        pop R18 ; POP r18 from stack
        pop R17 ; POP r17 from stack
        ret
```

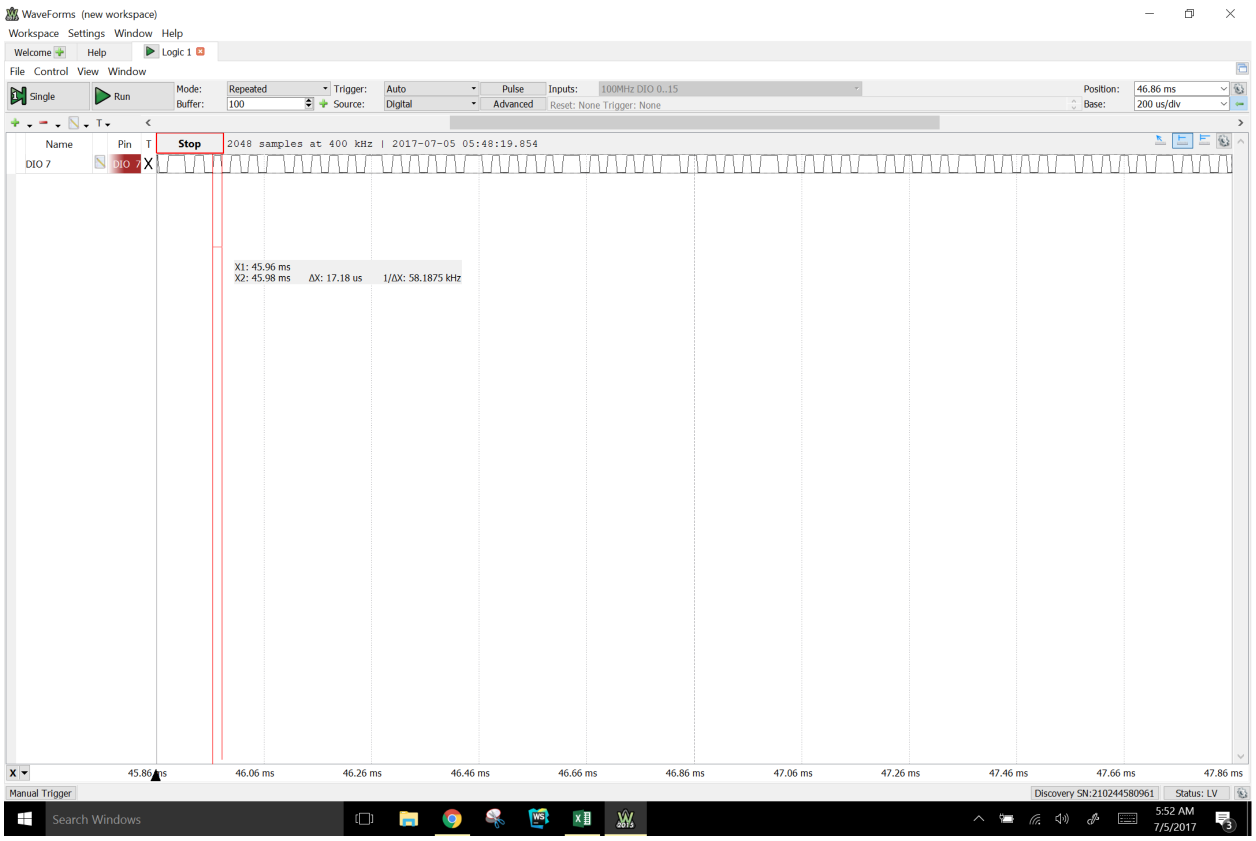
Code for lab4_serial_int.asm:

```
; Lab 4 Part E
; Name:           Michael Arboleda
; Section:        7F34
; TA Name:        Wesley Piard
; Description: Revices/Transmit using interrupts
;
; lab4_serial_int.asm
; Created: 7/5/2017 12:22:26 AM

.include "ATxmega128A1Udef.inc"

; address equates

; Constant equates
.equ new_clock_freq = 0b00000010
.equ TxRx_On = 0b00011000        ; 0x18
.equ pin_Tx  = 0b00001000        ; 0x08
.equ pin_Rx  = 0b00000100        ; 0x04
; asynch, 8 databits, odd parity, 1 start, and 1 stop
.equ usart_ctrl_C       = 0b00110011
; BSEL 576000
.equ upper_BSEL         = 0b00000100
.equ BSEL                         = 0b00110111
.equ BSCALE                       = 0b10110000    ; -5  BSCALE
.equ NULL                         = 0x00          ; Null character
.equ CR                           = 0x0D          ; Carriage Return
.equ LF                           = 0x0A          ; Line Feed
.equ low_int_lvl        = 0b00010000    ; Low level interupt config
.equ PMIC_crtl_lvl_config = 0b00000111  ;
.equ clk_div            = 0b00000111    ; DIV1024
;Color
.equ BIT456 = 0x70
.equ WHITE = ~(BIT456)

; Reg Defs
.def char_out = R1

;ORG defs
.ORG 0x0000                      ;Code starts running from address 0x0000.
        rjmp MAIN                ;Relative jump to start of program.

.org USARTD0_RXC_vect
        jmp USART_ISR
```

```
.org 0x0200

Test_String:
.db "Test_for_OUT_STRING", NULL

MAIN:

        call Change_CLK_32HZ     ; Change Clk to 32MHx
        call USART_INIT                    ; Initilize USART

        ldi R16, PMIC_crtl_lvl_config    ; LOAD PMIC lvl config
        sts PMIC_CTRL, R16                ; Set PMIC lvl config

        sei


        ; Set PORT D/LED to output
        ldi R16, BIT456 ;load a four bit value (PORTD is only four bits)
        sts PORTD_DIRSET, R16    ;set all the GPIO's in the four bit
                                         ; PORTD as outputs
        call Counter_INIT

; infinite loop
Never_End:
        lds R17, TCF0_CNT; LOAD the counter value
        lds R18, (TCF0_CNT + 1)

        cpi R18, 0x00            ; Compare counter (higher) value with 0
        brne Never_END           ; If not 0, restart loop
        cpi R17, 0x00            ; Compare counter (lower) value with 0
        brne Never_End           ; if not 0, resart loop
        ; Turn light on
        ldi R18, BIT456          ; LOAD white LED config
        sts PORTD_OUTTGL, R18    ; Output LED config

        rjmp Never_End           ; Jump to restart output loop




;*******************SUBROUTINES************************************
; Subroutine Name:  Counter_INIT
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Counter_INIT:
        push R16        ; PUSH r16
        push R17        ; PUSH r17
        push R18        ; PUSH r18

        ; Set up Counter 3D09
        ldi R16, 0x09           ; LOAD 0x09 into R16
        ldi R17, 0x3D           ; LOAD 0x3D into R17
        ldi R18, clk_div        ; LOAD Clk prescaler into R18

        sts TCF0_PER, R16                   ; Set Lower Bits of TOP
        sts (TCF0_PER + 1), R17             ; Set Higher Bits of TOP

        sts TCF0_CTRLA, R18                 ; Set Prescalar for Counter

        pop R18         ; POP r18
        pop R17         ; POP r17
        pop R16         ; POP r16
        ret



;********************Interupt Service Routine***************
; Subroutine Name:  USART_ISR
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
USART_ISR:
        push R16                                            ; Push R16

        lds r1, USARTD0_DATA                ; LOAD R1 with USART data
        lds R16, USARTD0_STATUS             ; LOAD R16 with USART STATUS reg
        CBR  r16, 7                         ; Clear interrupt flag at bit 7
        sts USARTD0_STATUS, R16 ; Store R16 into USART STATUS reg
        call OUT_CHAR                       ; Transmit char

        pop R16         ; Pop R16
        reti    ; return


;*******************SUBROUTINES***********************************
; Subroutine Name:  USART_INIT
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
USART_INIT:
        push R16                                    ; Push R16

        ldi R16, pin_Tx                     ; Pin3 config
        sts PortD_DIRSET, R16     ; Set Pin3 to transmit
        sts PortD_OUTSET, R16     ; Output High to Pin3


        ldi R16, pin_Rx                     ; Pin2 config
        sts PORTD_DIRCLR, R16     ; Set Pin2 to recieve

        ; Set Control A
        ldi R16, low_int_lvl      ; LOAD r16 with interrupt config
        sts USARTD0_CTRLA, R16    ; Set USART interrupt

        ; Set Control B
        ldi R16, TxRx_On                            ; LOAD Tx and Rx config
        sts USARTD0_CTRLB, R16              ; Set Tx, Rx lines

        ; Set Control C
        ldi R16, usart_ctrl_C               ; LOAD CTRL config
        sts USARTD0_CTRLC, R16              ; Set USART

        ; Set Baud Rate Ctrl B
        ldi R16, (BSCALE | upper_BSEL) ; OR BSCALE with 11:8 bit of BSEL
        sts USARTD0_BAUDCTRLB, R16          ; Set BAUD CTRL B

        ; Set Baud Rate Ctrl A
        ldi R16, BSEL                       ; LOAD 7:0 of BSEL in  R16
        sts USARTD0_BAUDCTRLA, R16                  ; Set BAUD CTRL A



        pop R16           ; POP r16
        ret               ; Return



;******************SUBROUTINES***********************************
; Subroutine Name:  OUT_CHAR
; Inputs: R1
; Outputs: No direct outputs
; Affected: None
OUT_CHAR:
        push R16                                          ; Push R16
```

```
Transfer_Complete:
        lds R16, USARTD0_STATUS         ; LOAD status reg to R16
        sbrs R16, 5             ; Skip jump if bit 5 is 1
        rjmp Transfer_Complete          ; Restart Loop
        sts USARTD0_DATA, R1            ; Output char thru USART

        pop R16         ; POP R16
        ret             ; Return

;*******************SUBROUTINES***********************************
; Subroutine Name: OUT_STRING
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
OUT_STRING:
        push R16        ; Push R16

While_String:
        lpm R16, Z+             ; LOAD Z data and increment pointer
        cpi R16, NULL           ; Compare Data and Null char
        breq End_While_String   ; If data = null, branch to exit
        mov R1, R16                     ; move R16 Data to R1
        call OUT_CHAR                   ; Call OUT_CHAR
        jmp While_String                ; Restart Loop

End_While_String:

        pop r16         ; Pop R16
        ret                     ; Return


;*******************SUBROUTINES***********************************
; Subroutine Name:  IN_CHAR
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
IN_CHAR:

        push R16                                        ; PUSH R16

Recieve_Complete:
        lds  R16, USARTD0_STATUS        ; LOAD the status register
        sbrs R16, 7                     ; Skip jump if bit 7 is 1
        rjmp Recieve_Complete           ; Restart Loop
        lds  R1, USARTD0_DATA           ; LOAD data into R1
        pop R16         ; POP r16
        ret             ; Return



;*******************SUBROUTINES***********************************
; Subroutine Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
; Affected: None
Change_CLK_32HZ:
        ;Push Values
        push R17 ; PUSH r17 to stack
        push R18 ; PUSH r18 to stack
        push R19 ; PUSH r19 to stack
        push R20 ; PUSH r20 to stack

; Enable the new oscillator
ldi R16, new_clock_freq         ; Load R16 with the clk-freq config 0x02
sts OSC_CTRL, r16                       ; Set the clk config

;Wait for the right flag to be set in the OSC_STATUS reg
; While flag is not set
While_32_flag:
        lds R17, OSC_STATUS             ; Load Status Flag
        and R17, R16                    ; Bit-mask with 00000010
        cp R17, R16                     ; Compare Mask and Value
        brne While_32_flag              ; Restart loop if flag not set

; Write the IOREG signature to the CPU_CCP reg
        ldi R17, CCP_IOREG_gc   ; Load IOREG into R17
        sts CPU_CCP, R17                ; Store IOREG into CPU CCP

;Select the new clock source in the CLK_CTRL reg
        ldi R17, CLK_SCLKSEL_RC32M_gc; load 32 MHz internal osc config
        sts CLK_CTRL, R17               ; Store config in clk control

        ;Pop Values
        pop R20 ; POP r20 from stack
        pop R19 ; POP r19 from stack
        pop R18 ; POP r18 from stack
        pop R17 ; POP r17 from stack
        ret
```

i. Appendix



WaveForms (new workspace)

Mode: Repeated
Buffer: 100
Trigger: Auto
Source: Digital
Inputs: 100MHz DIO 0..15
Reset: None Trigger: None
Position: 46.86 ms
Base: 200 us/div

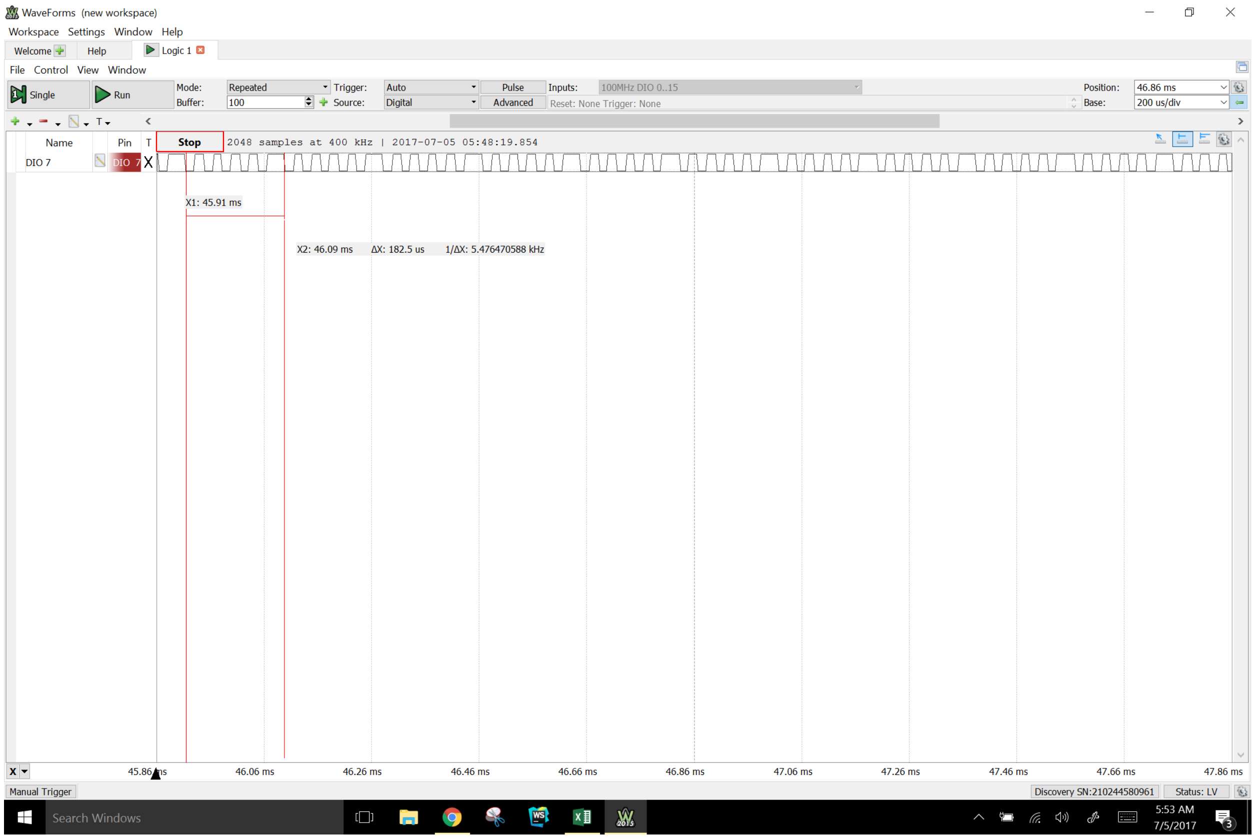
2048 samples at 400 kHz | 2017-07-05 05:48:19.854

X1: 45.96 ms
X2: 45.98 ms ΔX: 17.18 us 1/ΔX: 58.1875 kHz

Discovery SN:210244580961 Status: LV

Figure 1: Data for one bit

Using $\frac{1}{57600Hz} = 17.361\mu S$ we see that one bit should be around for 17.361 micro seconds. The screen shot shows a frequency of 58187.5HZ and 17.18 micro seconds for one bit. This is close to the theoretical.



Mode: Repeated
Buffer: 100
Trigger: Auto
Source: Digital
Inputs: 100MHz DIO 0..15
Reset: None Trigger: None
Position: 46.86 ms
Base: 200 us/div

Stop 2048 samples at 400 kHz | 2017-07-05 05:48:19.854

X1: 45.91 ms

X2: 46.09 ms ΔX: 182.5 us 1/ΔX: 5.476470588 kHz

Discovery SN:210244580961 Status: LV

Figure 2: Data for entire serial transmission

Since there are 11 bits, 1 for start, 8 for data, 1 for parity and 1 for stop, the time for the serial transmission should be $11 * 17.361\mu S = 190.971\mu S$. According to the screen shot it takes 182.5 μS, which is close to the theoretical