REQUIREMENTS NOT MET

N/A. All Requirements are met in this lab.

PROBLEMS ENCOUNTERED

Some problems encountered with part 1 of the lab includes the pre-lab question portion where it had to be researched heavily from the manuals and lecture resources to be completed. The problems encountered with part 2 was with figuring out the preset value of the configuration needed to be done. For part 3, the major issue was figuring out how to setup the USART module so that the serial output is readable by the DAD. This required the assistance of PI in explaining the necessary steps to be done. For part 4 to 5, it was completed with relatively little issues, but more problems arose in part 6 as I could not get the looping of the reading the input character to work, requiring me to redesign the necessary code for it to function property. Part 7, was completed through reviewing the previous interrupt lab.

FUTURE WORK/APPLICATIONS

This lab was a good introduction into the implementation and function of USART. This lab is to be the expansion of more complex assembly programs, able to give the users' another way to interact with the microprocessor, enabling the running of reactions to human responses by the main program. With the USART commands users are no longer restricted to the hardware only to preset inputs on their uPad or external I/O hardware like LEDs and switches but allow for simple interactions from the console by the user. Like the subroutine, the way I program is now changed to be inclusive of USART for more capability of my programs. If given more time, the code of the lab could have been more organized and have a much neater layout to further reduce the likelihood of mistakes and further enhance the understanding of the program. With more time a more compacted or efficient communication can be implemented. Additionally, I could have used better instructions to make the code run more efficiently or learn to write more complex programs.

PRE-LAB EXERCISES

Part 1: Introduction to USART

i. What is the maximum possible baud rate that you could use for asynchronous communication within the USART system of the ATxmega128A1U, if the microcontroller is configured for a clock frequency of 2 MHz and has double-speed mode disabled (i.e., CLK2X = 0)? Support your answer.

The maximum possible baud rate that you could use for asynchronous communication within the USART system of the ATxmega128A1U, if the microcontroller is configured for a clock frequency of 2 MHz and has double-speed mode disabled (i.e., CLK2X = 0) is found with the formula $f_{BAUD} \leq \frac{f_{PER}}{16} = \frac{2*10^6}{16} = 125000\ bps$.

ii. In the context of the USART system within the ATxmega128A1U, how many buffers are used for a transmitter? How about for a receiver? Additionally, for both transmitters and receivers, explain what buffering signifies, in terms of flexibility given to an application.

In the context of the USART system within the ATxmega128A1U, the transmitter uses a single write buffer, allows continuous data transmission without any delay between frames, while the receiver uses a two-level receiver buffer. Buffering is to set the amount of data is going to be stored in order to preload the required data right before it gets used, either to be transmitted or used after being received. This gives the application the flexibility to hold on to data in the buffer, for the needed processes to run, till it was right for the buffered data to be called on and used.

iii. If an asynchronous serial communication protocol of 8 data bits, one start bit, one stop bit, no parity, and baud rate of 1 MHz was chosen, calculate how many seconds it would take to transmit the ASCII character string “Dr. Schwartz saw seven slick slimy snakes slowly sliding southward.” (This string has 67 characters.) Support your answer.

8 data bits + 1 start bit + 1 stop bit = 10 bits 67 characters = frames
67 ~~frames~~ * 10 ~~bits~~/~~frame~~ * 1 sec/(1*10^6 ~~bits~~) = 0.67 sec
It would take 0.67 sec to transmit the ASCII character string.

PSEUDOCODE/FLOWCHARTS

SECTION X (1, 2, etc.)

Part 2: USART, Character Transmission

Johnny Li
Lab 5

Part 2 Flow chart

Main:
- Equate number
- Set registers
- rcall USART_INIT
- rcall OUT_CHAR
- rcall Loop

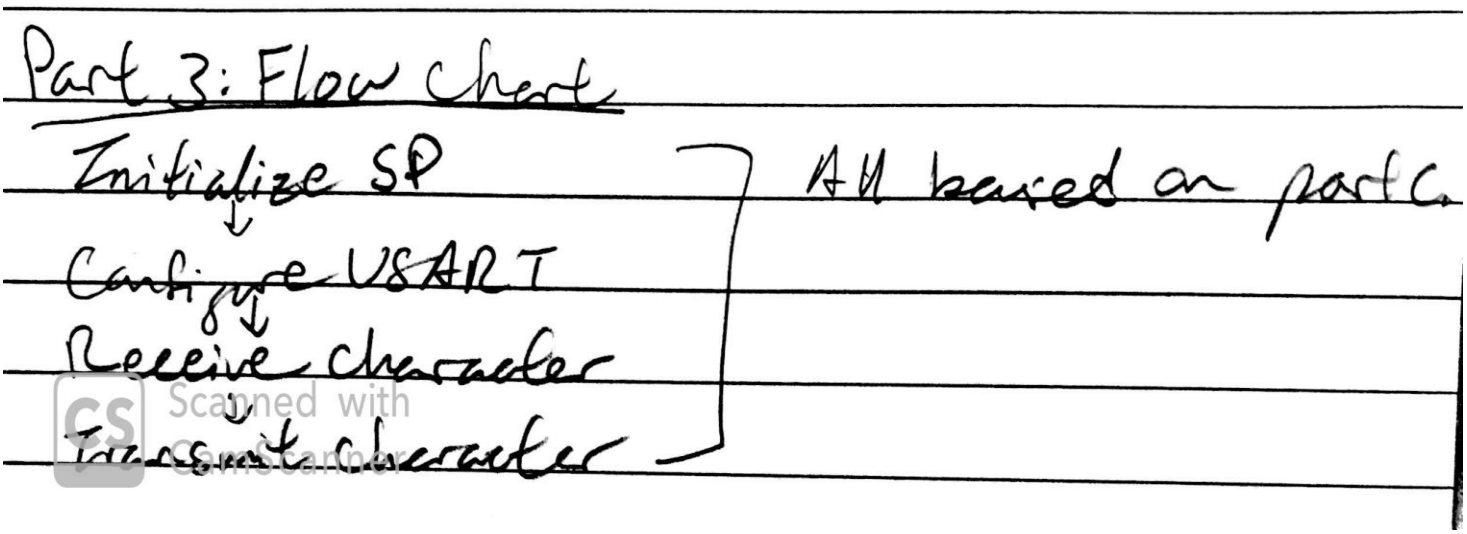
Loop:
- Loop to output U.
- rjmp Loop

USART_INIT:
- Set data direction on pin.
- Set mode, amount of data bits and type.
- Set baud rate. (57,600 bps)
- ret

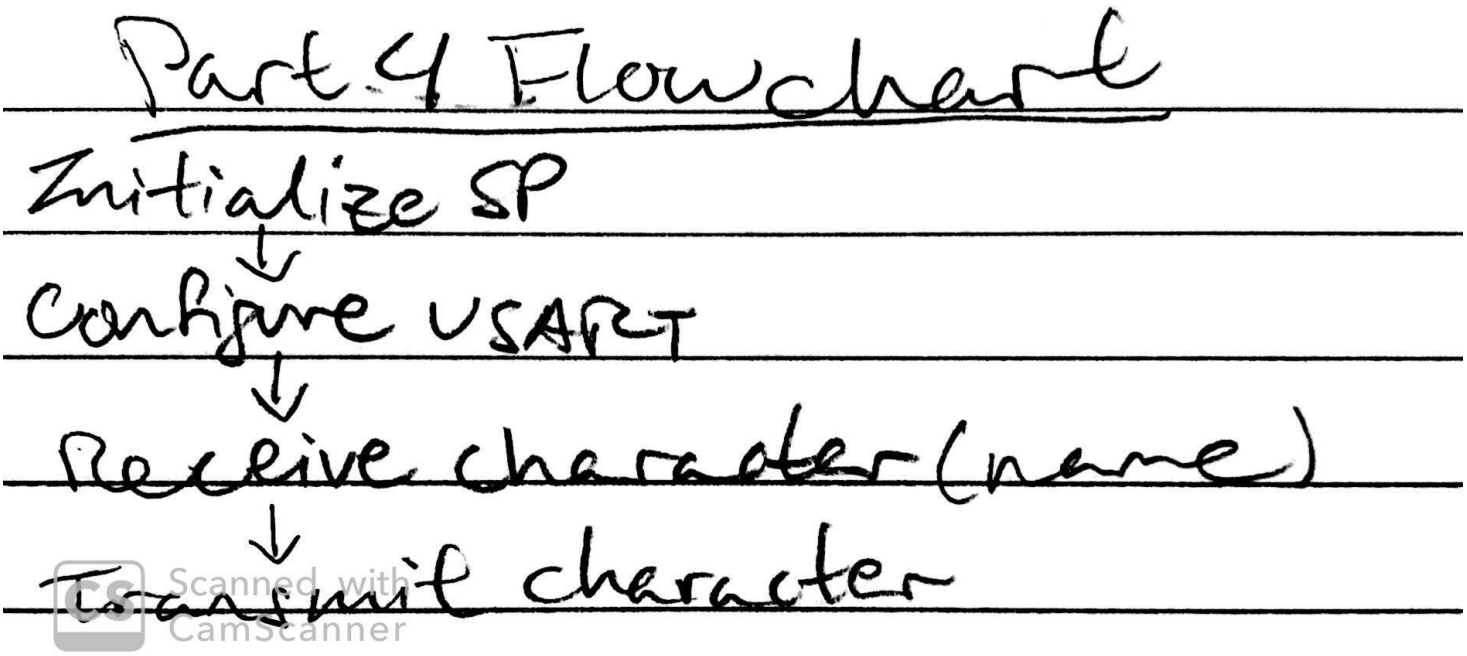
8 data
1 start
1 stop
even parity

OUT_CHAR:
- Output a character to transmit pin, by r16.
- Check if there is a transmission, poll till its over.
- ret

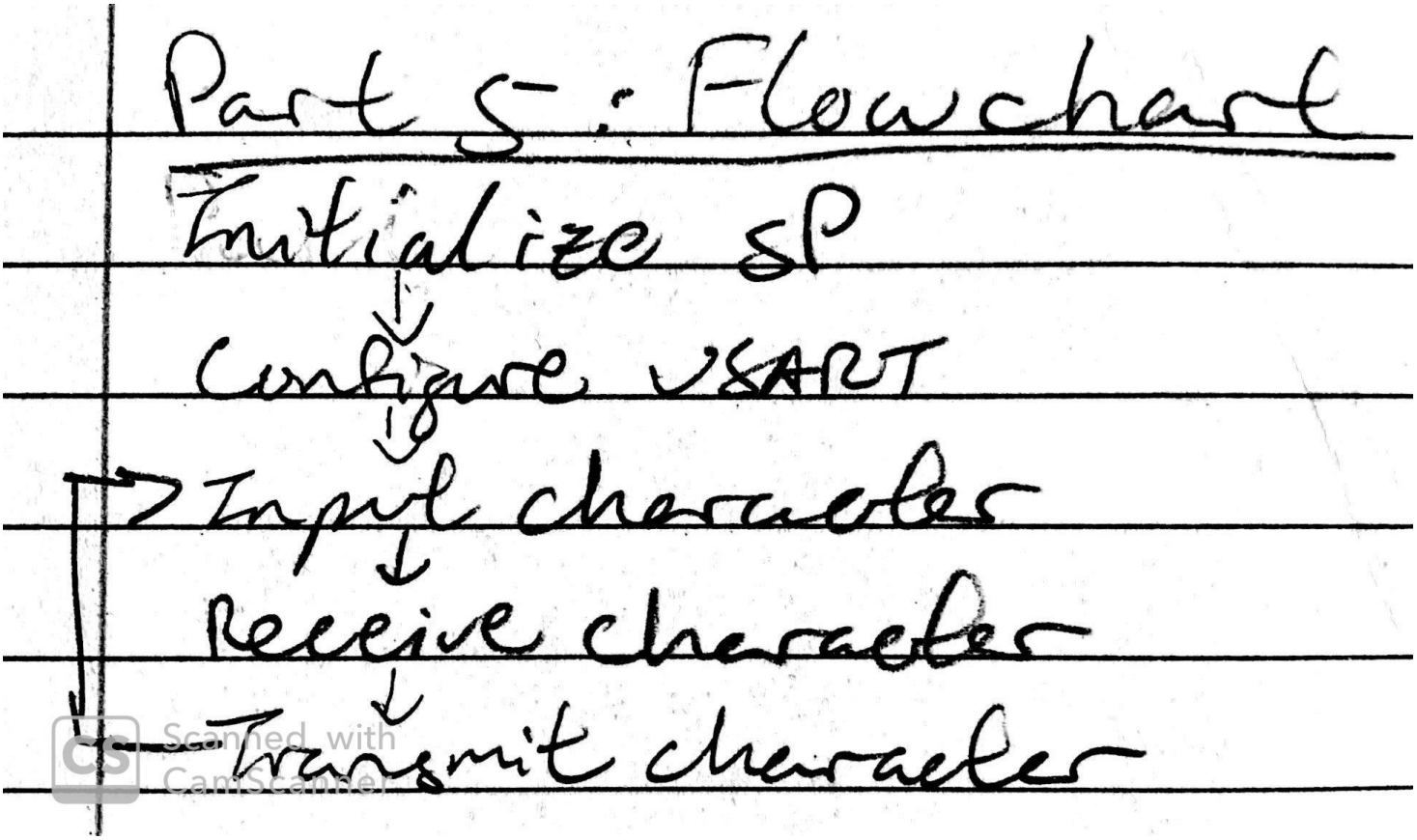
Part 3: USART, Measuring Baud Rate



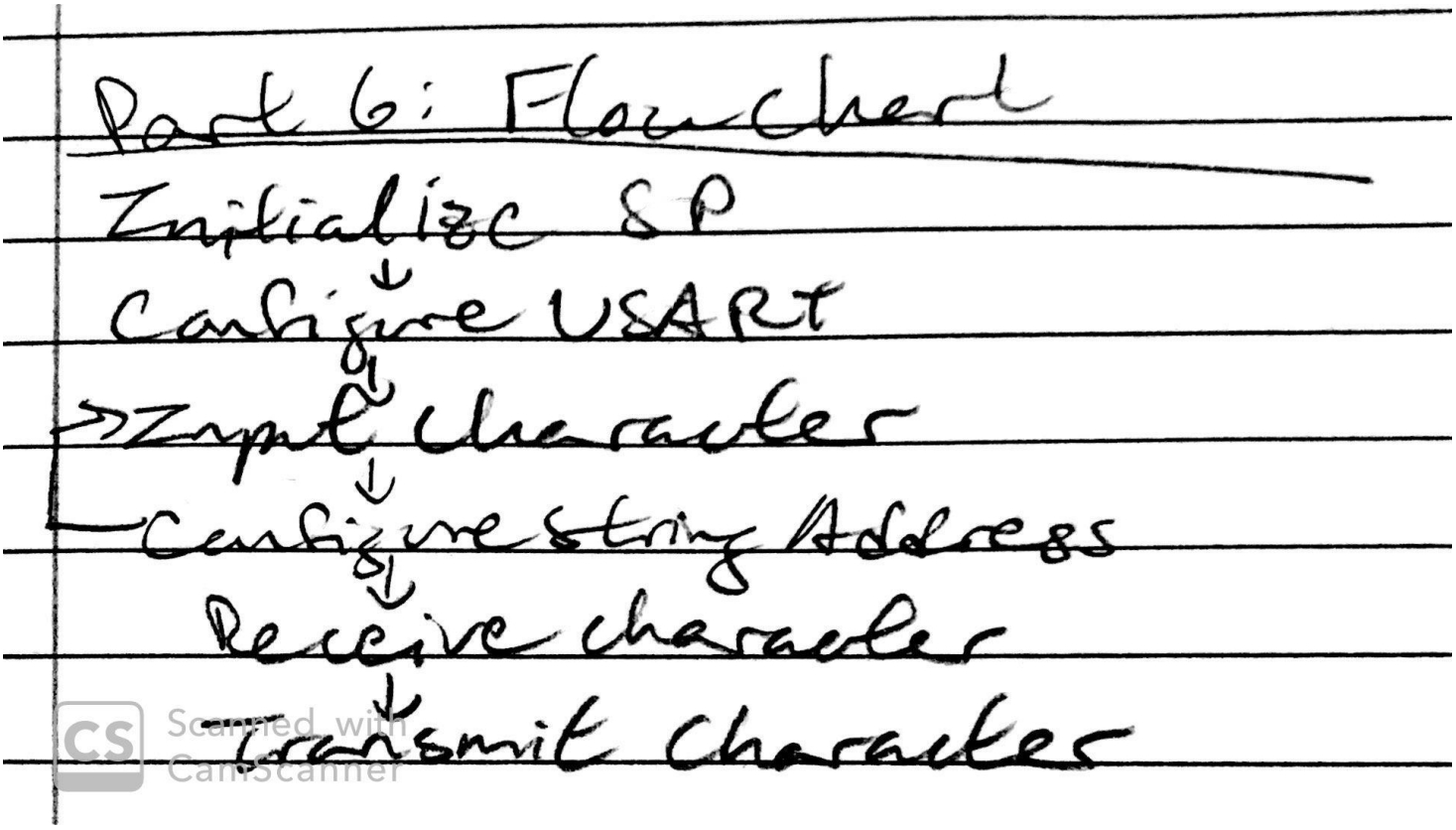
Part 4: USART, String Transmission



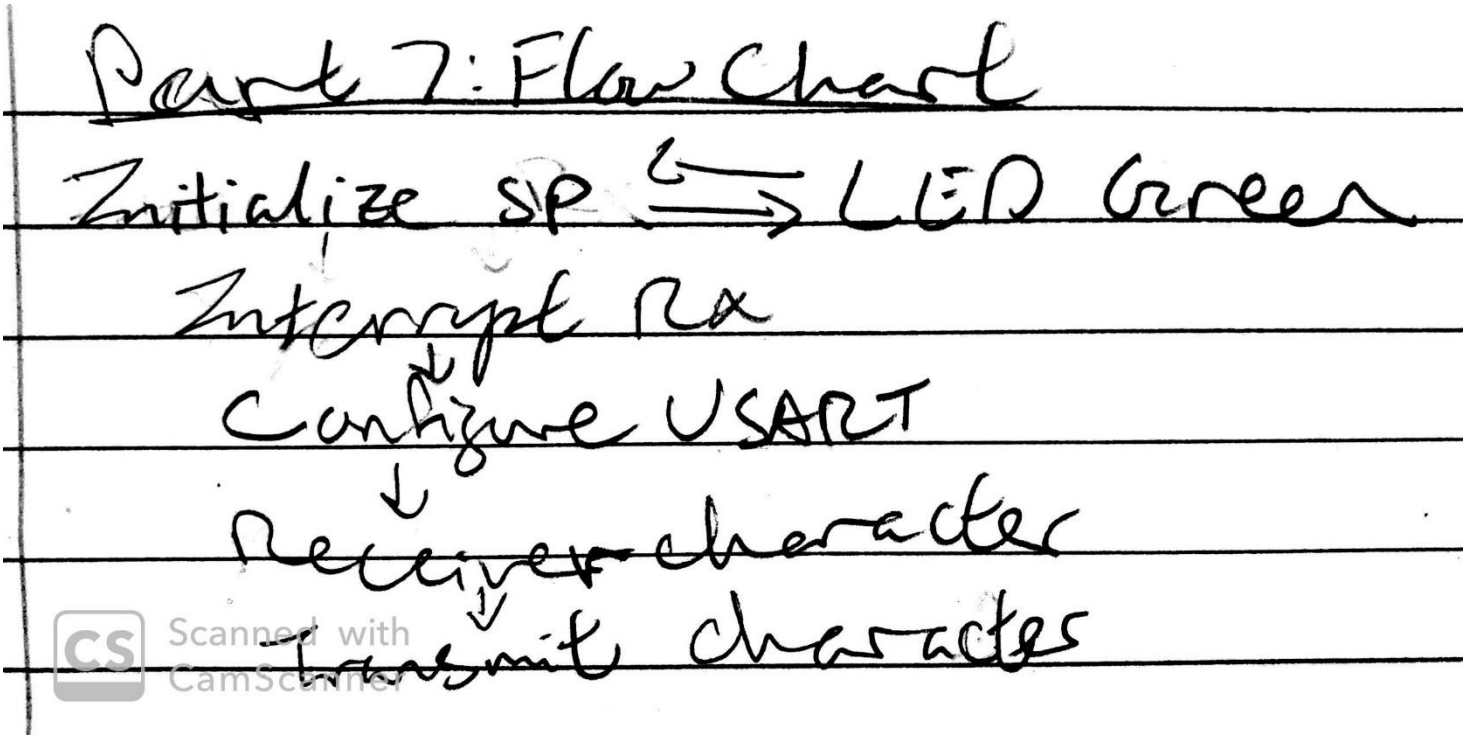
Part 5: USART, Character Input



Part 6: USART, String Input



Part 7: USART, Interrupt-Based Receiving



PROGRAM CODE

SECTION X (1, 2, etc.)

Part 2: USART, Character Transmission

```
;*****************************************************************************
;Lab 5 Part 2
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, CHARACTER TRANSMISSION
;*********************************INCLUDES************************************
;*********************************MAIN PROGRAM********************************
	.cseg
.org 0x0000
	rjmp MAIN

.org 0x100
MAIN:
	ldi YL, 0xFF  ;initialize low byte of stack pointer
	out CPU_SPL, YL
	ldi YL, 0x3F
	out CPU_SPH, YL

	rcall INIT_USART

	nop
LOOP:
	ldi r16, 'U'
	rcall OUT_CHAR
	rjmp LOOP

;*********************************SUBROUTINES*********************************
; NAME:                      INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               56000 (115200) BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;*****************************************************************************
INIT_USART:
	push r16

	ldi R16, pin_Tx
	sts PortD_OUTSET, R16	;set the TX line to default to '1' as
                                        ;  described in the documentation
	sts PortD_DIRSET, R16	;Must set PortD_PIN3 as output for TX pin
                                        ;  of USARTD0
	ldi R16, pin_Rx
	sts PortD_DIRCLR, R16	;Set RX pin for input

	ldi R16, TR_xON                         ;INIT_USART initializes UART 0 on PortD (PortD0)
	sts USARTD0_CTRLB, R16            ;Turn on TXEN, RXEN lines

	ldi R16, usart
```

```
       sts USARTD0_CTRLC, R16            ;Set Parity to none, 8 bit frame, 1 stop bit

       ldi R16, (BSel & 0xFF)            ;select only the lower 8 bits of BSel
       sts USARTD0_BAUDCTRLA, R16  ;set baudctrla to lower 8 bites 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
                                                  ;  and upper 4 bits are the BScale.
       pop r16
       ret
; ****************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:   OUT_CHAR
; FUNCTION:     Outputs the character in register R16 to the SCI Tx pin
;               after checking if the DREIF (Data register empty flag)
;                    is empty.  The PC terminal program will take this
;               received data and  put it on the computer screen.
; INPUT:        Data to be transmitted is in register R16.
; OUTPUT:       Transmit the data.
; DESTROYS:     None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None.
OUT_CHAR:
       push R17
POLL:
       lds R17, USARTD0_STATUS           ;load status register
       sbrs R17, 5                              ;proceed to writing out the char if
                                                  ;  the DREIF flag is set
       rjmp POLL                         ;else go back to polling
       sts USARTD0_DATA, R16             ;send the character out over the USART

       pop R17
       ret
;*****************************END OF SUBROUTINES*****************************
```

Part 3: USART, Measuring Baud Rate

```
;*****************************************************************************
;Lab 5 Part 3
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, Measuring Baud Rate
;*********************************MAIN PROGRAM*********************************
       .cseg
.org 0x0000
       rjmp MAIN

.org 0x100
MAIN:
       ldi YL, 0xFF  ;initialize low byte of stack pointer
       out CPU_SPL, YL
       ldi YL, 0x3F
       out CPU_SPH, YL

       rcall INIT_USART

       nop
```

```
LOOP:
	ldi r16, 'U'
	rcall OUT_CHAR
	rjmp LOOP

;*******************************SUBROUTINES***********************************
; NAME:                      INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               56000 (115200) BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;*****************************************************************************
INIT_USART:
	push r16

	ldi R16, pin_Tx
	sts PortC_OUTSET, R16	;set the TX line to default to '1' as
                                        ;  described in the documentation
	sts PortC_DIRSET, R16	;Must set PortC_PIN3 as output for TX pin
                                        ;  of USARTD0
	ldi R16, pin_Rx
	sts PortC_DIRCLR, R16	;Set RX pin for input

	ldi R16, TR_xON                         ;INIT_USART initializes UART 0 on PortD (PortD0)
	sts USARTC0_CTRLB, R16             ;Turn on TXEN, RXEN lines

	ldi R16, usart
	sts USARTC0_CTRLC, R16             ;Set Parity to none, 8 bit frame, 1 stop bit

	ldi R16, (BSel & 0xFF)             ;select only the lower 8 bits of BSel
	sts USARTC0_BAUDCTRLA, R16  ;set baudctrla to lower 8 bites of BSel

	ldi R16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)
	sts USARTC0_BAUDCTRLB, R16  ;set baudctrlb to BScale | BSel. Lower
                                                ;  4 bits are upper 4 bits of BSel
                                                ;  and upper 4 bits are the BScale.
	pop r16
	ret
; ****************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:   OUT_CHAR
; FUNCTION:     Outputs the character in register R16 to the SCI Tx pin
;               after checking if the DREIF (Data register empty flag)
;                    is empty.  The PC terminal program will take this
;               received data and  put it on the computer screen.
; INPUT:        Data to be transmitted is in register R16.
; OUTPUT:       Transmit the data.
; DESTROYS:     None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None.
OUT_CHAR:
	push R17
POLL:
	lds R17, USARTC0_STATUS           ;load status register
	sbrs R17, 5                             ;proceed to writing out the char if
                                                ;  the DREIF flag is set
```

```
        rjmp POLL                       ;else go back to polling
        sts USARTC0_DATA, R16           ;send the character out over the USART

        pop R17
        ret
;*****************************END OF SUBROUTINES*******************************
```

Part 4: USART, String Transmission

```
;***************************************************************************
;Lab 5 Part 4
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, String Transmission
;*******************************MAIN PROGRAM**********************************
        .cseg
.org 0x0000
        rjmp MAIN

String:
        .db "Johnny_Li", 0

.org 0x100
MAIN:
        ldi YL, 0xFF  ;initialize low byte of stack pointer
        out CPU_SPL, YL
        ldi YL, 0x3F
        out CPU_SPH, YL

        rcall INIT_USART

        ldi ZL, low(String<<1)
        ldi ZL, high(String<<1)
        rcall OUT_STRING

        nop
LOOP:
        //ldi r16, 'U'
        //rcall OUT_CHAR
        rjmp LOOP

;*********************************SUBROUTINES*********************************
; NAME:                      INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               56000 (115200) BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;***************************************************************************
INIT_USART:
        push r16

        ldi R16, pin_Tx
        sts PortD_OUTSET, R16       ;set the TX line to default to '1' as
                                            ;  described in the documentation
        sts PortD_DIRSET, R16       ;Must set PortD_PIN3 as output for TX pin
                                            ;  of USARTD0
```

```
	ldi R16, pin_Rx
	sts PortD_DIRCLR, R16	;Set RX pin for input

	ldi R16, TR_xON				;INIT_USART initializes UART 0 on PortD (PortD0)
	sts USARTD0_CTRLB, R16		;Turn on TXEN, RXEN lines

	ldi R16, usart
	sts USARTD0_CTRLC, R16		;Set Parity to none, 8 bit frame, 1 stop bit

	ldi R16, (BSel & 0xFF)		;select only the lower 8 bits of BSel
	sts USARTD0_BAUDCTRLA, R16  ;set baudctrla to lower 8 bites 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
                                ;  and upper 4 bits are the BScale.
	pop r16
	ret
;***************************************************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:   OUT_CHAR
; FUNCTION:     Outputs the character in register R16 to the SCI Tx pin
;               after checking if the DREIF (Data register empty flag)
;                    is empty.  The PC terminal program will take this
;               received data and  put it on the computer screen.
; INPUT:        Data to be transmitted is in register R16.
; OUTPUT:       Transmit the data.
; DESTROYS:     None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None.
OUT_CHAR:
	push R17
POLL:
	lds R17, USARTD0_STATUS		;load status register
	sbrs R17, 5					;proceed to writing out the char if
                                ;  the DREIF flag is set
	rjmp POLL					;else go back to polling
	sts USARTD0_DATA, R16		;send the character out over the USART

	pop R17
	ret
;***************************************************************************
; NAME:                     OUT_STRING
; FUNCTION:     Output a character string stored in program memory.
; INPUT:        None
; OUTPUT:       None
;***************************************************************************
OUT_STRING:
	push r16
Write:
	lpm r16, z+		;LOAD Z data and increment
	cpi r16, 0	;Check if null is reached
	breq End	;If null, reach end
	rcall OUT_CHAR
	rjmp Write

End:
	pop r16
	ret
;***************************END OF SUBROUTINES*****************************
```

Part 5: USART, Character Input

```
;***************************************************************************
;Lab 5 Part 5
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, Character Input
;*******************************MAIN PROGRAM**********************************
	.cseg
.org 0x0000
	rjmp MAIN

String:
	.db "Johnny_Li", 0

.org 0x100
MAIN:
	ldi YL, 0xFF  ;initialize low byte of stack pointer
	out CPU_SPL, YL
	ldi YL, 0x3F
	out CPU_SPH, YL

	rcall INIT_USART

	ldi ZL, low(String<<1)
	ldi ZL, high(String<<1)
	//rcall OUT_STRING
	rcall IN_CHAR

	nop
LOOP:
	//ldi r16, 'U'
	rcall OUT_CHAR
	rjmp LOOP

;*******************************SUBROUTINES***********************************
; NAME:                      INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               56000 (115200) BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;***************************************************************************
INIT_USART:
	push r16

	ldi R16, pin_Tx
	sts PortD_OUTSET, R16       ;set the TX line to default to '1' as
                                    ;  described in the documentation
	sts PortD_DIRSET, R16       ;Must set PortD_PIN3 as output for TX pin
                                    ;  of USARTD0
	ldi R16, pin_Rx
	sts PortD_DIRCLR, R16       ;Set RX pin for input

	ldi R16, TR_xON                     ;INIT_USART initializes UART 0 on PortD (PortD0)
	sts USARTD0_CTRLB, R16            ;Turn on TXEN, RXEN lines
```

```
	ldi R16, usart
	sts USARTD0_CTRLC, R16		;Set Parity to none, 8 bit frame, 1 stop bit

	ldi R16, (BSel & 0xFF)		;select only the lower 8 bits of BSel
	sts USARTD0_BAUDCTRLA, R16  ;set baudctrla to lower 8 bites 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
                                                ;  and upper 4 bits are the BScale.
	pop r16
	ret
;***************************************************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:   OUT_CHAR
; FUNCTION:     Outputs the character in register R16 to the SCI Tx pin
;               after checking if the DREIF (Data register empty flag)
;                    is empty.  The PC terminal program will take this
;               received data and  put it on the computer screen.
; INPUT:        Data to be transmitted is in register R16.
; OUTPUT:       Transmit the data.
; DESTROYS:     None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None.
OUT_CHAR:
	push R17
POLL:
	lds R17, USARTD0_STATUS		;load status register
	sbrs R17, 5                         ;proceed to writing out the char if
                                                ;  the DREIF flag is set
	rjmp POLL			;else go back to polling
	sts USARTD0_DATA, R16		;send the character out over the USART

	pop R17
	ret
;***************************************************************************
; NAME:                     OUT_STRING
; FUNCTION:     Output a character string stored in program memory.
; INPUT:        None
; OUTPUT:       None
;***************************************************************************
OUT_STRING:
	push r16
Read:
	lpm r16, z+		;LOAD Z data and increment
	cpi r16, 0    ;Check if null is reached
	breq End      ;If null, reach end
	rcall OUT_CHAR
	rjmp Read

End:
	pop r16
	ret
; ****************************************
; IN_CHAR polls the receive complete flag and will
;   pass the received character pack to the calling routine in R16.
; SUBROUTINE:   IN_CHAR
; FUNCTION:     Receives typed character (sent by the PC terminal
;               program through the PC to the PortD0 USART Rx pin)
;               into register R16.
```

```
; INPUT:        None.
; OUTPUT:       Register R16 = input from SCI
; DESTROYS:     R16 (result is transferred in this register)
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None
IN_CHAR:

Write:
       lds R16, USARTD0_STATUS           ;load the status register
       sbrs R16, 7                             ;proceed to reading in a char if
                                                ;  the receive flag is set
       rjmp Write                        ;else continue polling
       lds R16, USARTD0_DATA             ;read the character into R16

       ret
;*****************************END OF SUBROUTINES*******************************
```

Part 6: USART, String Input

```
;***************************************************************************
;Lab 5 Part 6
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, String Input
;*********************************MAIN PROGRAM*********************************
       .cseg
.org 0x0000
       rjmp MAIN

String:
       .db "Johnny_Li", 0

.org 0x100
MAIN:
       ldi YL, 0xFF  ;initialize low byte of stack pointer
       out CPU_SPL, YL
       ldi YL, 0x3F
       out CPU_SPH, YL

       rcall INIT_USART

       ;Input point
       ldi YL, low(0x2000)
       ldi YH, high(0x2000)

       ;Name point
       ldi ZL, low(String<<1)
       ldi ZH, high(String<<1)
       //rcall OUT_STRING   ;part 4
       //rcall IN_CHAR      ;part 5
       rcall IN_STRING

       nop
LOOP:
       //ldi r16, 'U'       ;part 2
       //rcall OUT_CHAR     ;part 2
       rcall NEW_OUT_STRING ;part 6

       rjmp LOOP
```

```
;*******************************SUBROUTINES***********************************
; NAME:                     INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               57600 BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;*****************************************************************************
INIT_USART:
       push r16

       ldi R16, pin_Tx
       sts PortD_OUTSET, R16       ;set the TX line to default to '1' as
                                           ;  described in the documentation
       sts PortD_DIRSET, R16       ;Must set PortD_PIN3 as output for TX pin
                                           ;  of USARTD0
       ldi R16, pin_Rx
       sts PortD_DIRCLR, R16       ;Set RX pin for input

       ldi R16, TR_xON                     ;INIT_USART initializes UART 0 on PortD (PortD0)
       sts USARTD0_CTRLB, R16            ;Turn on TXEN, RXEN lines

       ldi R16, usart
       sts USARTD0_CTRLC, R16            ;Set Parity to none, 8 bit frame, 1 stop bit

       ldi R16, (BSel & 0xFF)            ;select only the lower 8 bits of BSel
       sts USARTD0_BAUDCTRLA, R16  ;set baudctrla to lower 8 bites 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
                                                  ;  and upper 4 bits are the BScale.
       pop r16
       ret
;*****************************************************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:   OUT_CHAR
; FUNCTION:     Outputs the character in register R16 to the SCI Tx pin
;               after checking if the DREIF (Data register empty flag)
;                    is empty.  The PC terminal program will take this
;               received data and  put it on the computer screen.
; INPUT:        Data to be transmitted is in register R16.
; OUTPUT:       Transmit the data.
; DESTROYS:     None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None.
OUT_CHAR:
       push R17
POLL:
       lds R17, USARTD0_STATUS           ;load status register
       sbrs R17, 5                             ;proceed to writing out the char if
                                                  ;  the DREIF flag is set
       rjmp POLL                         ;else go back to polling
       sts USARTD0_DATA, R16             ;send the character out over the USART

       pop R17
       ret
```

```
;***********************************************************************
; NAME:                     OUT_STRING
; FUNCTION:     Output a character string stored in program memory.
; INPUT:        None
; OUTPUT:       None
;***********************************************************************
OUT_STRING:
       push r16
Read:
       lpm r16, Z+          ;LOAD Z data and increment
       cpi r16, 0    ;Check if null is reached
       breq End      ;If null, reach end
       rcall OUT_CHAR
       rjmp Read

End:
       pop r16
       ret
; ****************************************
; IN_CHAR polls the receive complete flag and will
;   pass the received character pack to the calling routine in R16.
; SUBROUTINE:   IN_CHAR
; FUNCTION:     Receives typed character (sent by the PC terminal
;               program through the PC to the PortD0 USART Rx pin)
;               into register R16.
; INPUT:        None.
; OUTPUT:       Register R16 = input from SCI
; DESTROYS:     R16 (result is transferred in this register)
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None
IN_CHAR:

Write:
       lds R16, USARTD0_STATUS           ;load the status register
       sbrs R16, 7                             ;proceed to reading in a char if
                                                       ;  the receive flag is set
       rjmp Write                        ;else continue polling
       lds R16, USARTD0_DATA             ;read the character into R16

       ret

;***********************************************************************
; NAME:                     IN_STRING
; FUNCTION:     Input character string stored in program memory.
; INPUT:        None
; OUTPUT:       None
;***********************************************************************
IN_STRING:
       rcall IN_CHAR        ;read the character into R16
       rcall OUT_CHAR
       cpi r16,0x0D         ;check if new line
       brne Check
       ;Store the character in Y address
       ldi r20,1
       rjmp Finish

Check:
       cpi r16, 0x08
       breq Remove
       cpi r16, 0x7F
       breq Remove
```

```
	st Y+, r16
	rjmp IN_STRING

Remove:
	sbiw Y, 1
	rjmp IN_STRING

Finish:
	ldi r16, 0	;add null value
	st Y+, r16
	ret
;***************************************************************************
; NAME:                     NEW_OUT_STRING
; FUNCTION:     Output a character string stored in data memory from input.
; INPUT:        None
; OUTPUT:       None
;***************************************************************************
NEW_OUT_STRING:
	push r16
	;reset Y
	ldi YL, low(0x2000)
	ldi YH, high(0x2000)
Reading:
	ld r16, y+		;LOAD Z data and increment
	cpi r16, 0	;Check if null is reached
	breq Ending	;If null, reach end
	rcall OUT_CHAR
	rjmp Reading

Ending:
	pop r16
	ret
;*****************************END OF SUBROUTINES*********************************
```

Part 7: USART, Interrupt-Based Receiving

```
;***************************************************************************
;Lab 5 Part 7
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, Interrupt-Based Receiving
;*********************************INCLUDES**************************************
;*******************************MAIN PROGRAM************************************
	.cseg
.org USARTD0_RXC_vect
	rjmp ISR

.org 0x0000
	rjmp MAIN

String:
	.db "Johnny_Li", 0

.org 0x100
MAIN:
	ldi YL, 0xFF  ;initialize low byte of stack pointer
	out CPU_SPL, YL
	ldi YL, 0x3F
	out CPU_SPH, YL
```

```
	rcall INIT_USART

	;Input point
	ldi YL, low(0x2000)
	ldi YH, high(0x2000)

	;Name point
	ldi ZL, low(String<<1)
	ldi ZH, high(String<<1)
	//rcall OUT_STRING   ;part 4
	//rcall IN_CHAR      ;part 5
	//rcall IN_STRING    ;part 6

	;GREEN_PWM
	ldi r16, 0x05
	sts PORTD_DIRCLR, r16       ;set portD initally off

	ldi r16, 0b00100000  ;load 1 to register LED
	sts PORTD_DIRSET,r16 ;set portD as output

	nop
LOOP:
	//ldi r16, 'U'       ;part 2
	//rcall OUT_CHAR     ;part 2
	//rcall NEW_OUT_STRING      ;part 6

	;Turn GREEN off
	ldi r16, 0xFF
	sts PORTD_OUT, r16
	;Turn BLUE on
	ldi r16, 0x00
	sts PORTD_OUT, r16
	;Loop endless
	rjmp LOOP

;*******************************SUBROUTINES***********************************
; NAME:                     INIT_USART
; FUNCTION:     Initializes the USARTDO's TX and Rx,
;               57600 BAUD, 8 data bits, 1 stop bit.
; INPUT:        None
; OUTPUT:       None
; DESTROYS:     R16
; REGS USED:  USARTD0_CTRLB, USARTD0_CTRLC, USARTD0_BAUDCTRLA,
;               USARTD0_BAUDCTRLB
; CALLS:        None.
;*****************************************************************************
INIT_USART:
	push r16

	ldi R16, pin_Tx
	sts PortD_OUTSET, R16       ;set the TX line to default to '1' as
                                        ;  described in the documentation
	sts PortD_DIRSET, R16       ;Must set PortD_PIN3 as output for TX pin
                                        ;  of USARTD0
	ldi R16, pin_Rx
	sts PortD_DIRCLR, R16       ;Set RX pin for input

	ldi R16, TR_xON                         ;INIT_USART initializes UART 0 on PortD (PortD0)
	sts USARTD0_CTRLB, R16            ;Turn on TXEN, RXEN lines

	ldi R16, usart
	sts USARTD0_CTRLC, R16            ;Set Parity to none, 8 bit frame, 1 stop bit
```

```
	ldi R16, (BSel & 0xFF)			;select only the lower 8 bits of BSel
	sts USARTD0_BAUDCTRLA, R16	;set baudctrla to lower 8 bites 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
							;  and upper 4 bits are the BScale.

	ldi r16, 0b00010000		;interrupt enable
	sts USARTD0_CTRLA, r16

	ldi r16, 0x01
	sts PMIC_CTRL, r16	;low level interrupt

	sei
	pop r16
	ret
;***************************************************************************
; OUT_CHAR receives a character via R16 and will
;   poll the DREIF (Data register empty flag) until it true,
;   when the character will then be sent to the USART data register.
; SUBROUTINE:	OUT_CHAR
; FUNCTION:	Outputs the character in register R16 to the SCI Tx pin
;		after checking if the DREIF (Data register empty flag)
;		    is empty.  The PC terminal program will take this
;		received data and  put it on the computer screen.
; INPUT:	Data to be transmitted is in register R16.
; OUTPUT:	Transmit the data.
; DESTROYS:	None.
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:	None.
OUT_CHAR:
	push R17
POLL:
	lds R17, USARTD0_STATUS			;load status register
	sbrs R17, 5				;proceed to writing out the char if
							;  the DREIF flag is set
	rjmp POLL				;else go back to polling
	sts USARTD0_DATA, R16			;send the character out over the USART

	pop R17
	ret
;***************************************************************************
; NAME:			OUT_STRING
; FUNCTION:	Output a character string stored in program memory.
; INPUT:	None
; OUTPUT:	None
;***************************************************************************
OUT_STRING:
	push r16
Read:
	lpm r16, Z+		;LOAD Z data and increment
	cpi r16, 0	;Check if null is reached
	breq End	;If null, reach end
	rcall OUT_CHAR
	rjmp Read

End:
	pop r16
	ret
; ****************************************
```

```
; IN_CHAR polls the receive complete flag and will
;   pass the received character pack to the calling routine in R16.
; SUBROUTINE:   IN_CHAR
; FUNCTION:     Receives typed character (sent by the PC terminal
;               program through the PC to the PortD0 USART Rx pin)
;               into register R16.
; INPUT:        None.
; OUTPUT:       Register R16 = input from SCI
; DESTROYS:     R16 (result is transferred in this register)
; REGS USED:  USARTD0_STATUS, USARTD0_DATA
; CALLS:        None
IN_CHAR:

Write:
       lds R16, USARTD0_STATUS           ;load the status register
       sbrs R16, 7                              ;proceed to reading in a char if
                                                        ;  the receive flag is set
       rjmp Write                        ;else continue polling
       lds R16, USARTD0_DATA             ;read the character into R16

       ret

;***************************************************************************
; NAME:                     IN_STRING
; FUNCTION:     Input character string stored in program memory.
; INPUT:        None
; OUTPUT:       None
;***************************************************************************
IN_STRING:
       rcall IN_CHAR        ;read the character into R16
       rcall OUT_CHAR
       cpi r16,0x0D         ;check if new line
       brne Check
       ;Store the character in Y address
       ldi r20,1
       rjmp Finish

Check:
       cpi r16, 0x08
       breq Remove
       cpi r16, 0x7F
       breq Remove

       st Y+, r16
       rjmp IN_STRING

Remove:
       sbiw Y, 1
       rjmp IN_STRING

Finish:
       ldi r16, 0    ;add null value
       st Y+, r16
       ret
;***************************************************************************
; NAME:                     NEW_OUT_STRING
; FUNCTION:     Output a character string stored in data memory from input.
; INPUT:        None
; OUTPUT:       None
;***************************************************************************
NEW_OUT_STRING:
       push r16
```

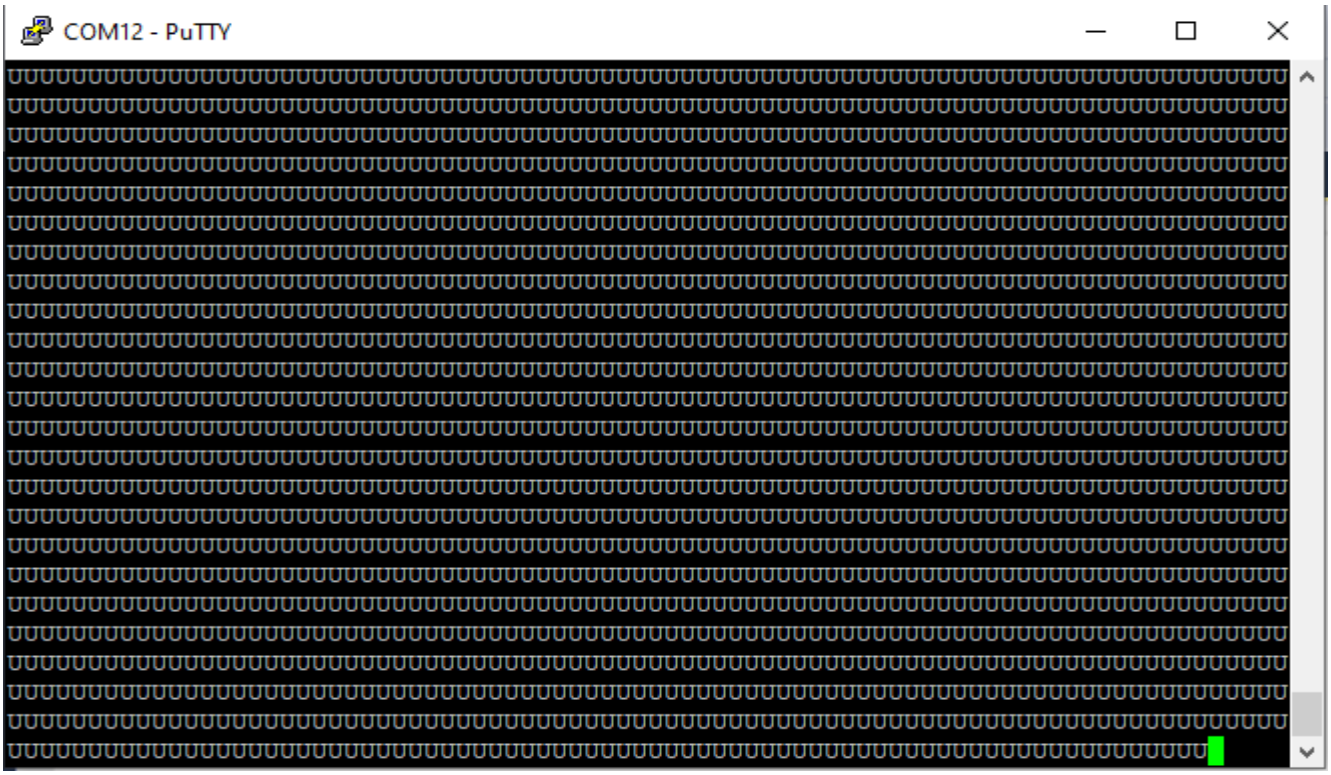
```
	;reset Y
	ldi YL, low(0x2000)
	ldi YH, high(0x2000)
Reading:
	ld r16, y+		;LOAD Z data and increment
	cpi r16, 0	;Check if null is reached
	breq Ending	;If null, reach end
	rcall OUT_CHAR
	rjmp Reading

Ending:
	pop r16
	ret
;***************************************************************************
; NAME:			ISR
; FUNCTION:	Interrupt USART module to echo, i.e., re-transmit, any character
;			received by your microcontroller back to your computer.
; INPUT:		None
; OUTPUT:		None
;***************************************************************************
ISR:
	in r19, CPU_SREG
	push r19

	lds r20, USARTD0_DATA	;load input
	sts USARTD0_DATA, r20	;put input to output console terminal

	pop r19
	out CPU_SREG, r19
	reti
;***************************END OF SUBROUTINES*****************************
```

APPENDIX

Part 2: USART, Character Transmission

```
;*****************************************************************************
;Lab 5 Part 2
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, CHARACTER TRANSMISSION
;***********************************INCLUDES************************************
;***********************************INCLUDES************************************
.include "ATxmega128a1udef.inc"
;*******************************END OF INCLUDES*********************************
;*******************************DEFINED SYMBOLS*********************************
.equ TR_xON = 0b00011000    ;0x18
.equ pin_Tx = 0b00001000    ;0x08
.equ pin_Rx = 0b00000100    ;0x04
.equ usart = 0b00100011     ;asynch, 8 data, even, 1 stat, 1 stop
.EQU BSel = 9
.EQU BScale = -3    ;57600 Hz
;***************************END OF DEFINED SYMBOLS******************************
;*******************************MEMORY CONSTANTS********************************
;***************************END OF MEMORY CONSTANTS*****************************
```

Screenshot 1: Part 2-USART Memory Configuration

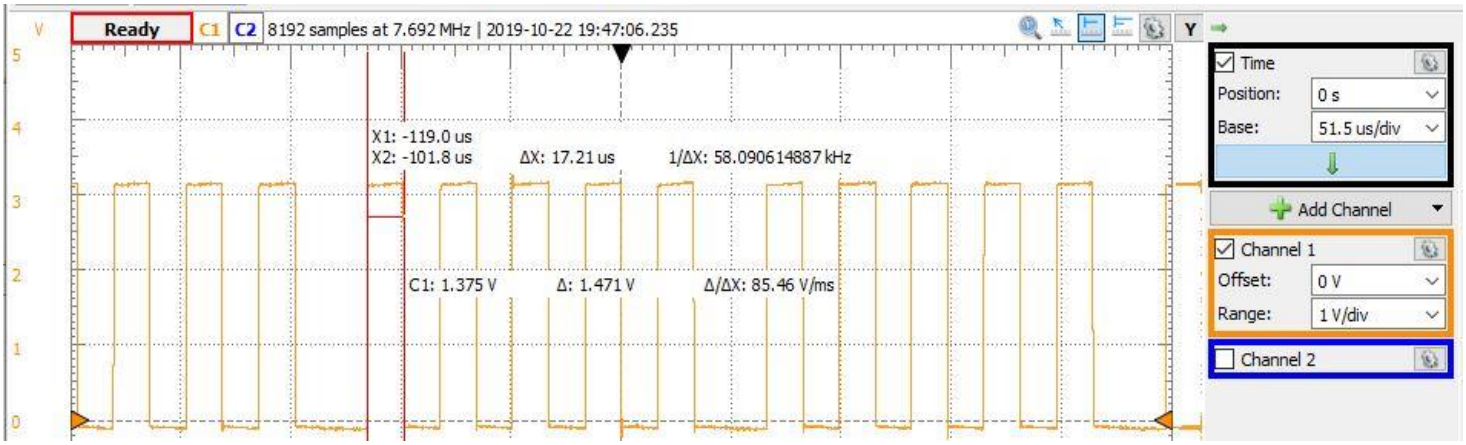


Screenshot 2: All ‘U’ Output on Command Console

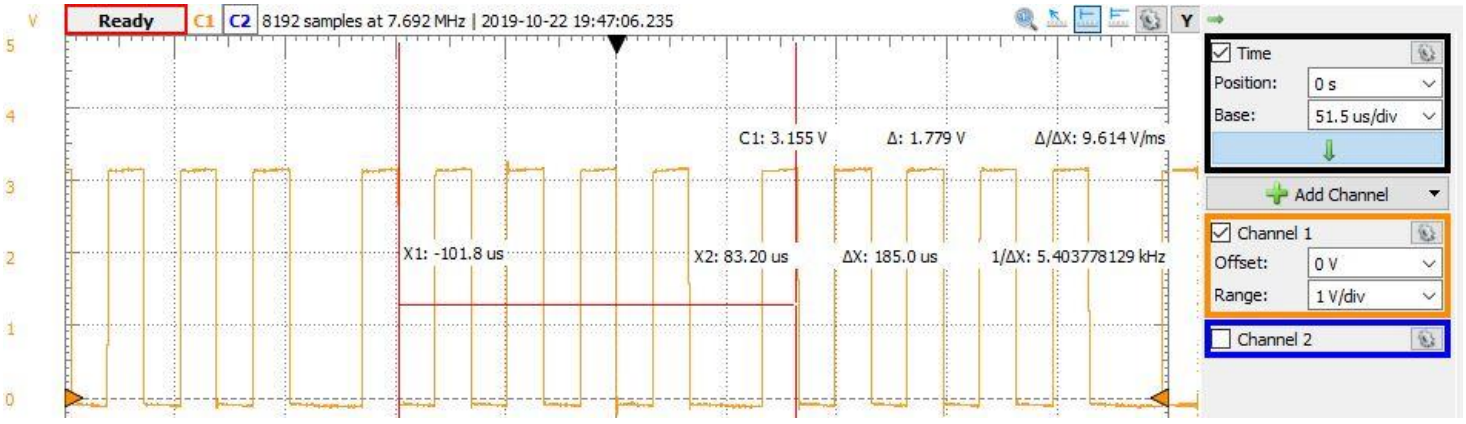
Part 3: USART, Measuring Baud Rate

```
;*********************************************************************************
;Lab 5 Part 3
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, Measuring Baud Rate
;***********************************INCLUDES**************************************
;***********************************INCLUDES**************************************
.include "ATxmega128a1udef.inc"
;*****************************END OF INCLUDES*************************************
;*****************************DEFINED SYMBOLS*************************************
.equ TR_xON = 0b00011000    ;0x18
.equ pin_Tx = 0b00001000    ;0x08
.equ pin_Rx = 0b00000100    ;0x04
.equ usart = 0b00100011     ;asynch, 8 data, even, 1 stat, 1 stop
.equ BSel = 9
.equ BScale = -3    ;57600 Hz
;Port C (J2) pin 3 to read serial transmission.
;***************************END OF DEFINED SYMBOLS********************************
;*****************************MEMORY CONSTANTS************************************
;***************************END OF MEMORY CONSTANTS*******************************
```

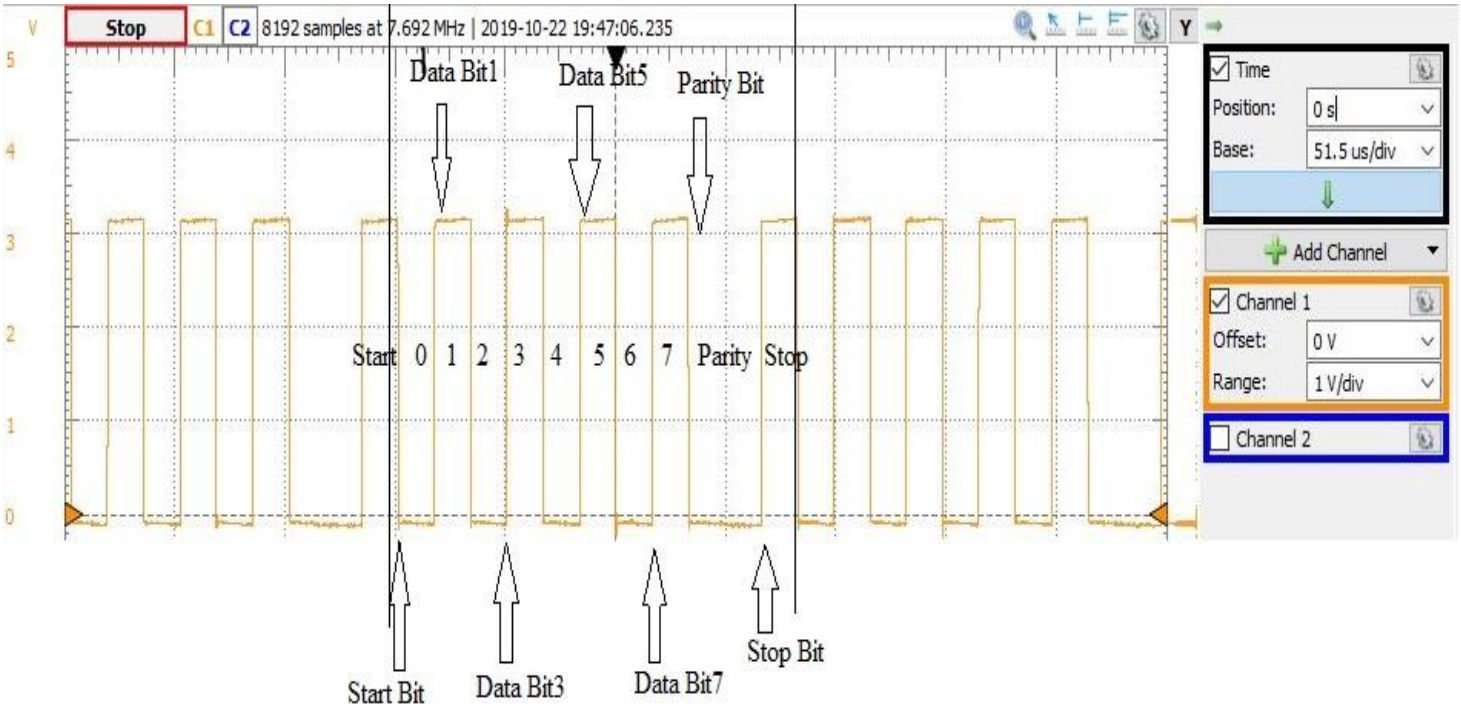
Screenshot 3: Part 3-USART measure Memory Configuration



Screenshot 4: DAD measure the width of both a single data bit.



Screenshot 5: DAD measure the single transmission frame.



Screenshot 6: ID the single transmission frame.

Part 4: USART, String Transmission

```
;Lab 5 Part 4
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, String Transmission
;***********************************INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;*******************************END OF INCLUDES*********************************
;*******************************DEFINED SYMBOLS*********************************
.equ TR_xON = 0b00011000    ;0x18
.equ pin_Tx = 0b00001000    ;0x08
.equ pin_Rx = 0b00000100    ;0x04
.equ usart = 0b00100011     ;asynch, 8 data, even, 1 stat, 1 stop
.equ BSel = 9
.equ BScale = -3    ;57600 Hz
;***************************END OF DEFINED SYMBOLS*******************************
;*******************************MEMORY CONSTANTS*********************************
;***************************END OF MEMORY CONSTANTS******************************
```

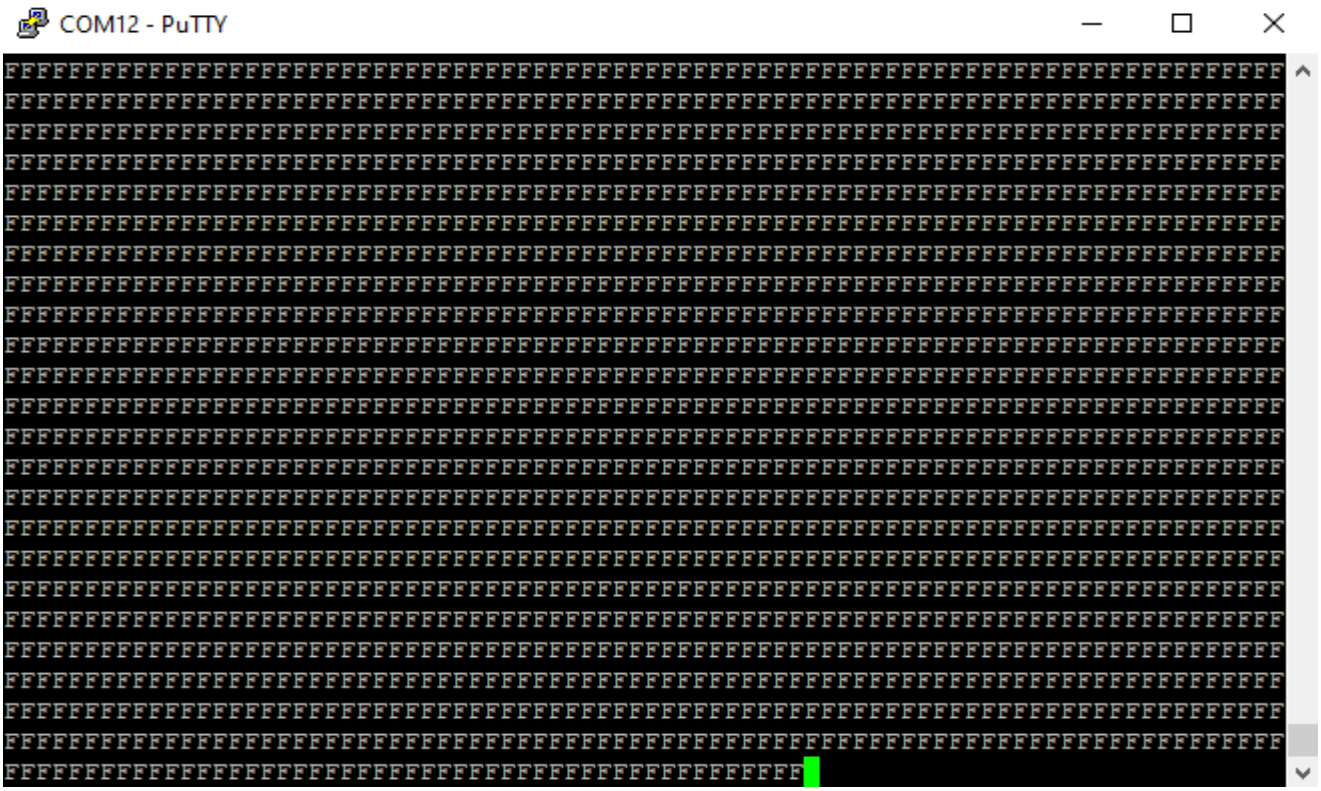
Screenshot 7: Part 4-USART String Memory Configuration

Screenshot 8: Name Output on Command Console

Part 5: USART, Character Input

```
1  ;*****************************************************************************
2  ;Lab 5 Part 5
3  ;Section #: 1823
4  ;Name: Johnny Li
5  ;Class #: 12378
6  ;PI Name: Jared Holley
7  ;Description: USART, Character Input
8  ;*********************************INCLUDES*************************************
9  .include "ATxmega128a1udef.inc"
10 ;*****************************END OF INCLUDES*********************************
11 ;*****************************DEFINED SYMBOLS*********************************
12 .equ TR_xON = 0b00011000     ;0x18
13 .equ pin_Tx = 0b00001000     ;0x08
14 .equ pin_Rx = 0b00000100     ;0x04
15 .equ usart = 0b00100011      ;asynch, 8 data, even, 1 stat, 1 stop
16 .equ BSel = 9
17 .equ BScale = -3     ;57600 Hz
18 ;*************************END OF DEFINED SYMBOLS******************************
19 ;*****************************MEMORY CONSTANTS********************************
20 ;*************************END OF MEMORY CONSTANTS*****************************
```

Screenshot 9: Part 5-USART Input Memory Configuration

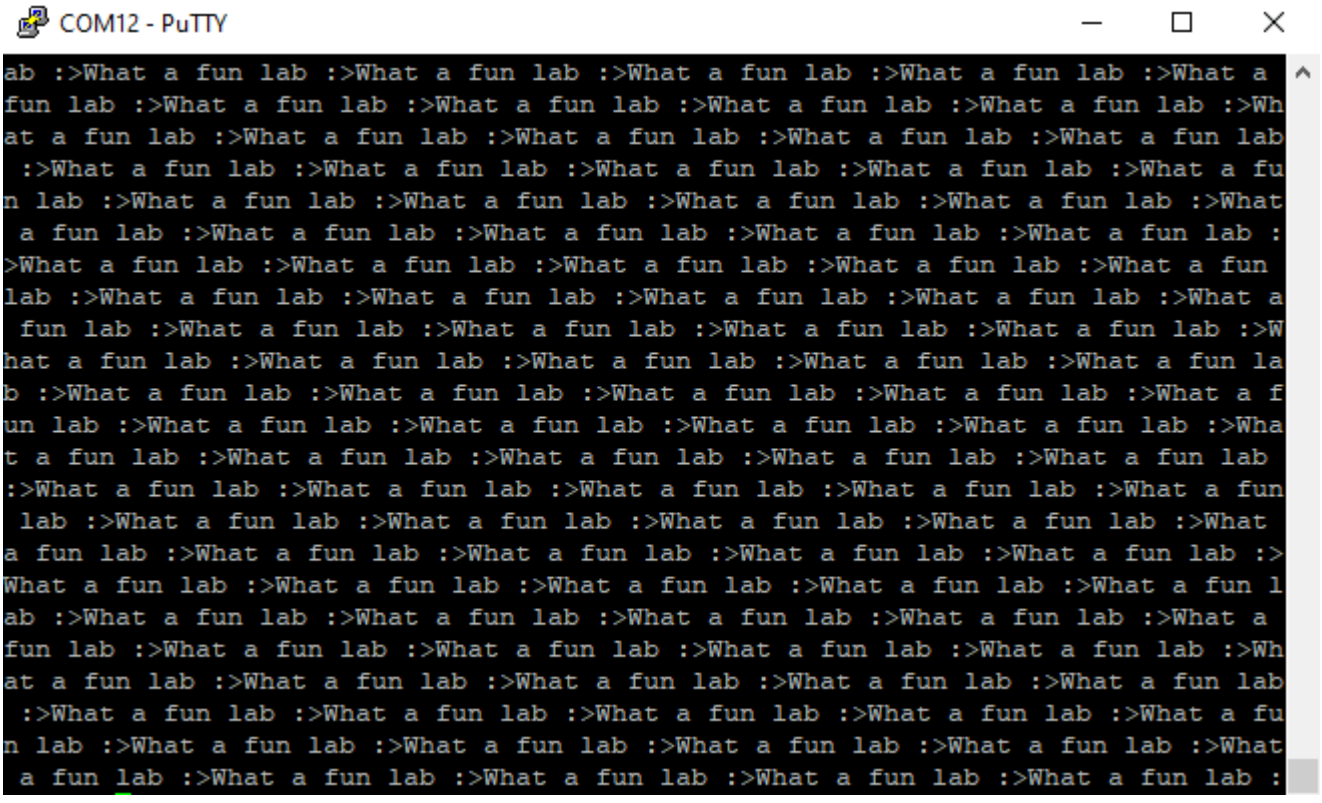


Screenshot 10: Input character Output on Command Console

Part 6: USART, String Input

```
1  ;***************************************************************************
2  ;Lab 5 Part 6
3  ;Section #: 1823
4  ;Name: Johnny Li
5  ;Class #: 12378
6  ;PI Name: Jared Holley
7  ;Description: USART, String Input
8  ;***********************************INCLUDES************************************
9  .include "ATxmega128a1udef.inc"
10 ;*******************************END OF INCLUDES*********************************
11 ;*******************************DEFINED SYMBOLS*********************************
12 .equ TR_xON = 0b00011000    ;0x18
13 .equ pin_Tx = 0b00001000    ;0x08
14 .equ pin_Rx = 0b00000100    ;0x04
15 .equ usart = 0b00100011     ;asynch, 8 data, even, 1 stat, 1 stop
16 .equ BSel = 9
17 .equ BScale = -3    ;57600 Hz
18 ;***************************END OF DEFINED SYMBOLS*****************************
19 ;*******************************MEMORY CONSTANTS********************************
20 ;*****************************END OF MEMORY CONSTANTS***************************
```

Screenshot 11: Part 6-USART Input String Memory Configuration



Screenshot 12: Input string output on Command Console

Part 7: USART, Interrupt-Based Receiving

```
;*********************************************************************************
;Lab 5 Part 7
;Section #: 1823
;Name: Johnny Li
;Class #: 12378
;PI Name: Jared Holley
;Description: USART, Interrupt-Based Receiving
;***********************************INCLUDES*************************************
.include "ATxmega128a1udef.inc"
;*******************************END OF INCLUDES**********************************
;*******************************DEFINED SYMBOLS**********************************
.equ TR_xON = 0b00011000    ;0x18
.equ pin_Tx = 0b00001000    ;0x08
.equ pin_Rx = 0b00000100    ;0x04
.equ usart = 0b00100011     ;asynch, 8 data, even, 1 stat, 1 stop
.equ BSel = 9
.equ BScale = -3    ;57600 Hz
;Port C (J2) pin 3 to read serial transmission.
;PORT D to commuicate with the computer.
;***************************END OF DEFINED SYMBOLS*******************************
;*******************************MEMORY CONSTANTS*********************************
;***************************END OF MEMORY CONSTANTS******************************
```

Screenshot 13: Part 7-USART Interrupt Memory Configuration

Screenshot 14: Input string output on Command Console