REQUIREMENTS NOT MET

N/A

PROBLEMS ENCOUNTERED

I encountered issues when attempting to store and echo strings from both program and data memory. The problem was traced to incorrect pointer handling with the Z and Y registers, as well as the absence of a null terminator at the end of each string. Adjusting the LPM and LD/ST instructions and ensuring that a null byte was stored at the end of each received string resolved the issue and allowed for the proper output of stored data. My part 7 program initially failed because the USART Receive Complete interrupt never executed. After verifying the PMIC configuration and USART control registers, I realized I forgot to jump to the ISR at the USART_D0_RXC vector. All issues were successfully resolved, and full USART functionality was achieved.

FUTURE WORK/APPLICATIONS

With more time and resources, the USART communication system developed in this lab could be expanded for more practical applications. One possible improvement would be creating a command-based interface that allows a user to send specific instructions from a computer for the microcontroller to process. The program could also be adapted to send sensor data or system information to a PC for monitoring or data logging. Future versions could include support for multiple USART modules to allow communication between several devices at once. These extensions would turn the basic USART setup into a more versatile and powerful communication tool for embedded systems.

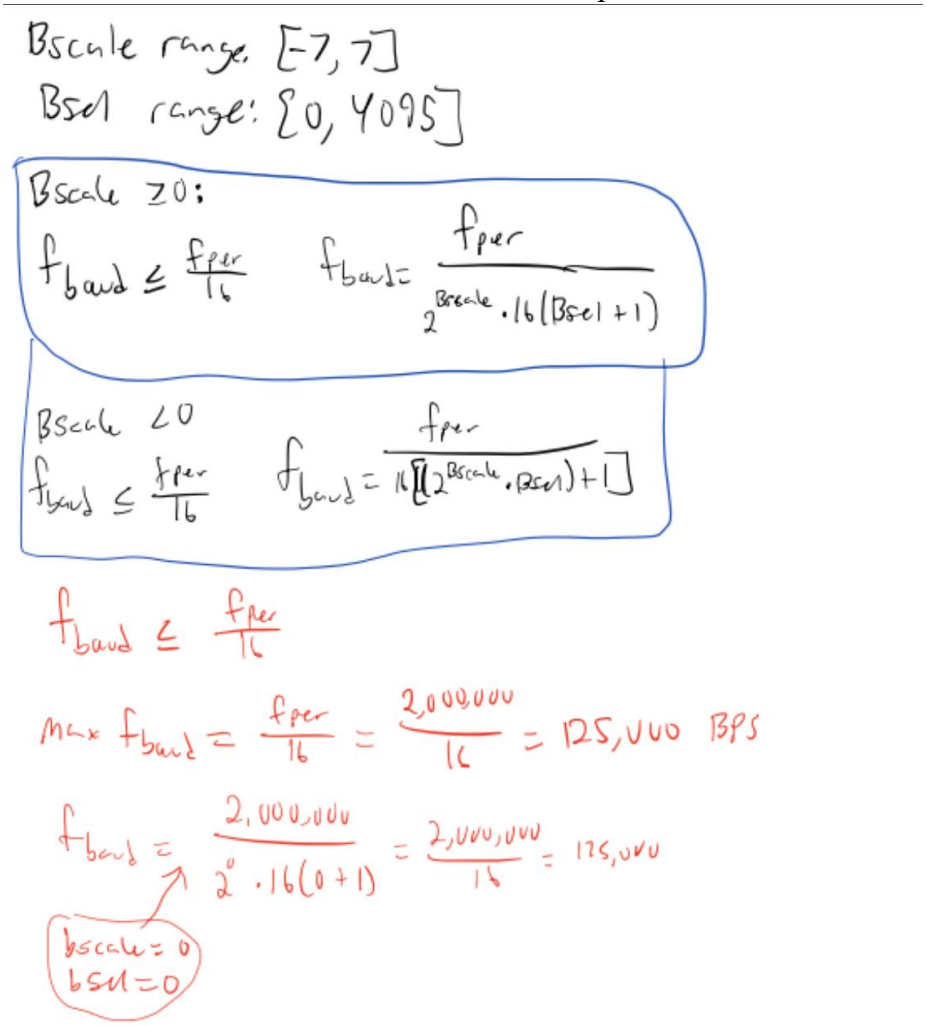
PRE-LAB EXERCISES

i. The sampling rate of a UART receiver is usually faster than the baud rate of the overall system. Why is this so?

   i. The sampling rate of a UART receiver is usually faster (16 times faster in our processor) than the baud rate of the overall system so that it can accurately detect each bit during transmission. This higher rate allows the receiver to detect the start bit edge more precisely and increases the likelihood that each bit is sampled near its midpoint, ensuring reliable data reception.

ii. What is the maximum possible baud rate for asynchronous communication within the USART system of the ATxmega128A1U, assuming that the microcontroller has a system clock frequency of 2 MHz and that the USART "double-speed mode" is disabled (i.e., the relevant bit CLK2X is set to 0)? In addition to the maximum rate, provide the values of the relevant registers used to configure that rate. Whenever appropriate, support your answer with calculations.

   i. The maximum possible baud rate for asynchronous communication within the USART system of the ATxmega128A1U, assuming that the microcontroller has a system clock frequency of 2 MHz and that the USART "double-speed mode" is disabled is 125, 000 bits per second since the Baud rate must be less than the Fper / 16.



1.

   ii. The values of the relevant registers used to configure this rate are:

1. USARTPx_BAUDCTRLA = 0x00
    a. Sets BSEL [7:0] to 0
2. USARTPx_BAUDCTRLB = 0x00
    a. Sets BSCALE to 0 and the upper four bits of BSEL to 0 also
3. USARTPx_CTRLB bit 2 = 0
    a. Sets the clk2x bit to 0 which keeps the clock in normal mode
4. These are all the registers required to set this rate, but others are required to fully configure the USART system.

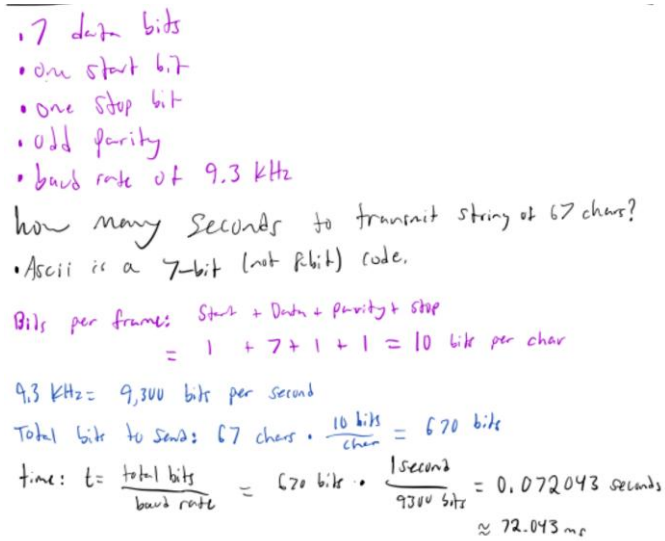
iii. In the context of the USART system within the ATxmega128A1U, how many buffers (i.e., memory locations that store temporary data) are used by a transmitter? How many are used by a receiver? Additionally, for both transmitters and receivers, explain how the use of buffers provides greater flexibility to an application involving these components.

i. In the context of the USART system within the ATxmega128A1U, two buffers are used by the transmitter and two buffers are used by the receiver.
    1. Transmit data register and transmit shift register
    2. Receiver data register and receiver shift register

ii. For both transmitters and receivers, the use of buffers provides greater flexibility to an application involving these components by allowing the CPU and USART hardware to work more independently. While one byte is being shifted out or in by the hardware, another byte can already be written to or read from the data register by the CPU. This overlap provides greater flexibility and efficiency, allowing continuous data flow in applications that require frequent communication.

iv. If an asynchronous serial communication protocol of 7 data bits, one start bit, one stop bits, odd parity, and baud rate of 9.3 kHz 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.) Note that ASCII is a 7-bit (not an 8-bit) code. Show all work.

i. If an asynchronous serial communication protocol of 7 data bits, one start bit, one stop bits, odd parity, and baud rate of 9.3 kHz was chosen it would take 0.072043 seconds or 72.043 ms.



1.

PSEUDOCODE/FLOWCHARTS

SECTION 1

Lab5_2.asm

```
*/

.include "Atxmega128A1Udef.inc"


.equ CR = 13

.equ LF = 10

.dseg

.org 0x2000


.cseg

.org 0x0000

        rjmp MAIN


.org 0x200

MAIN:

;init stack



;init USART PINS

rcall INIT_USART_PINS

;init USART

rcall  INIT_USART


REPEAT:


;load ‘U’
```

```
;OUT_CHAR expects the argument in r16

Rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTD0 TX pin) and
                pin 2 for input (PortD0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTD_PIN3 as output for TX pin
                        of USARTD0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTD_DIR, PORTD_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0

;set PORTD_PIN2 as input for rx of USARTd0

ret
```

```
;*********************************************
/* INIT_USART
init uart 0 on portd (PORTD0)
subroutine: INIT_USART
function:       init the usartd0's tx and rx
                        67000 Hz, 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:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale



;enable Tx and Rx

ret
```

```
;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
                        recieved 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

TX_POLL:
;load status register



;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling

;send the character out over the USART

pop r17
```

```
ret
```

Lab5_3.asm (same as part 2 except uses PORTC instead of PORTD)

```
*/
.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10
.dseg
.org 0x2000

.cseg
.org 0x0000
        rjmp MAIN

.org 0x200
MAIN:
;init stack



;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

REPEAT:

;load ‘U’
;OUT_CHAR expects the argument in r16
```

```
Rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTC0 TX pin) and
                    pin 2 for input (PortC0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTC_PIN3 as output for TX pin
                        of USARTC0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTC_DIR, PORTC_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portC0 tx is in bit 3
.equ BIT2 = 1<<2; portC0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTC_PIN3 as output for tx pin of USARTC0

;set PORTC_PIN2 as input for rx of USARTC0

ret



;*********************************************
```

```
/* INIT_USART
init uart 0 on portC (PORTC0)
subroutine: INIT_USART
function:       init the usartC0's tx and rx
                    67000 Hz, 8 data bits, 1 stop bit
INPUT: none
OUTPUT: none
Destroys: r16
regs used:      USARTC0_CTRLB, USARTC0_CTRLC, USARTC0_BAUDCTRLA,
                    USARTC0_BAUDCTRLB
calls: none
*/
INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale



;enable Tx and Rx

ret
```

```
;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
                        recieved 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:      USARTC0_STATUS, USARTC0_DATA
calls: none
*/

OUT_CHAR:
push r17

TX_POLL:
;load status register



;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling

;send the character out over the USART

pop r17

ret
```

Lab5_4.asm

```
*/
.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10
.equ String = 0x2000

.dseg

.cseg
.org 0x2000
String = “Arion Stern”

.org 0x0000
        rjmp MAIN

.org 0x200
MAIN:
;init stack

;init Z pointer to point to STRING

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART
```

```
rcall OUT_STRING

REPEAT:

Rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTD0 TX pin) and
                    pin 2 for input (PortD0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTD_PIN3 as output for TX pin
                        of USARTD0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTD_DIR, PORTD_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0

;set PORTD_PIN2 as input for rx of USARTd0
```

```
ret



;*********************************************
/* INIT_USART
init uart 0 on portd (PORTD0)
subroutine: INIT_USART
function:       init the usartd0's tx and rx
                        67000 Hz, 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:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale
```

```
;enable Tx and Rx


ret


;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
                        recieved 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


TX_POLL:
;load status register



;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling


;send the character out over the USART
```

```
pop r17


ret


/*************************************************************************************
* Name:     OUT_STRING
* Purpose:  output a character string stored in program memory
* Inputs:   none
* Outputs:  None
* Affected:
* Notes:
*************************************************************************************/
OUT_STRING:
;push registers
;read value pointed to by Z and increment z
;if the character is null jump to return
        ;else move its value into r16 and call OUT_CHAR
Rjmp OUT_STRING


RETURN:
;pop registers
ret
```

Lab5_5.asm

```
*/
```

```
.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10
.equ String = 0x2000

.dseg

.cseg
.org 0x2000
String = “Arion Stern”

.org 0x0000
        rjmp MAIN

.org 0x200
MAIN:
;init stack

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

REPEAT:

;read a character

;output that character
```

```
;Repeat
rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTD0 TX pin) and
                    pin 2 for input (PortD0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTD_PIN3 as output for TX pin
                        of USARTD0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTD_DIR, PORTD_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0

;set PORTD_PIN2 as input for rx of USARTd0

ret
```

```
;*********************************************
/* INIT_USART
init uart 0 on portd (PORTD0)
subroutine: INIT_USART
function:       init the usartd0's tx and rx
                        67000 Hz, 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:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale



;enable Tx and Rx

ret
```

```
;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
                        recieved 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

TX_POLL:
;load status register



;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling

;send the character out over the USART

pop r17
```

```
ret


/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  output a character string stored in program memory
* Inputs:   none
* Outputs:  None
* Affected:
* Notes:
*****************************************************************************/
OUT_STRING:
;push registers
;read value pointed to by Z and increment z
;if the character is null jump to return
        ;else move its value into r16 and call OUT_CHAR
Rjmp OUT_STRING


RETURN:
;pop registers
Ret
;**********************************************
/* IN_CHAR polls the receive complete flag and will
                pass the receivedchar back to the calling routine in r16
subroutine: IN_CHAR
function:       receives typed char (sent by PC terminal
                        program through the PC to the PORTD0 USART Rx pin)
                        into register r16.
INPUT: none
OUTPUT: r16 = input from SCI
Destroys: r16 (result is transferred in this register)
```

regs used: USARTD0_STATUS, USARTD0_DATA

calls: none

**********/

IN_CHAR:

RX_POLL:

;load status register

;check if the receive flag (RXCIF) is set (to read in a character)

;read the character into r16

ret

Lab5_6.asm

*/

.include "Atxmega128A1Udef.inc"

.equ CR = 13

.equ LF = 10

.equ BS = 0x08

.equ DEL = 0x7F

.equ String = 0x2000

.equ StringSize = 32

.dseg

.org 0x2000

INPUT_TAB:

.byte StringSize

```
.cseg
.org 0x2000
String = “Arion Stern”

.org 0x0000
        rjmp MAIN

.org 0x200
MAIN:
;init stack

;init Y pointer

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

;read the string from uart
rcall IN_STRING

;reset the Y pointer and echo

REPEAT:
rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTD0 TX pin) and
```

```
                pin 2 for input (PortD0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTD_PIN3 as output for TX pin
                        of USARTD0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTD_DIR, PORTD_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0

;set PORTD_PIN2 as input for rx of USARTd0

ret



;*********************************************
/* INIT_USART
init uart 0 on portd (PORTD0)
subroutine: INIT_USART
function:       init the usartd0's tx and rx
                        67000 Hz, 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:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale



;enable Tx and Rx

ret

;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
```

recieved 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

TX_POLL:

;load status register

;check if the data register empty flag (DREIF) is set (to send out a char)

;else go back to polling

;send the character out over the USART

pop r17

ret

/***

* Name: OUT_STRING

* Purpose: output a character string stored in program memory

* Inputs: none

* Outputs: None

* Affected:

```
* Notes:
*************************************************************************************/

OUT_STRING:

;push registers

;read value pointed to by Z and increment z

;if the character is null jump to return

        ;else move its value into r16 and call OUT_CHAR

Rjmp OUT_STRING


RETURN:

;pop registers

Ret

;*********************************************
/* IN_CHAR polls the receive complete flag and will

                pass the receivedchar back to the calling routine in r16

subroutine: IN_CHAR

function:       receives typed char (sent by PC terminal

                        program through the PC to the PORTD0 USART Rx pin)

                        into register r16.

INPUT: none

OUTPUT: r16 = input from SCI

Destroys: r16 (result is transferred in this register)

regs used:      USARTD0_STATUS, USARTD0_DATA

calls: none

*****************************************************/

IN_CHAR:


RX_POLL:

;load status register
```

```
;check if the receive flag (RXCIF) is set (to read in a character)

;read the character into r16

ret

/*****************************************************************************
* Name:     IN_STRING
* Purpose:  Receive string from USARTD0 into data memory via Y-pointer.
* Inputs:   Y = start address of buffer.
* Notes:    - Stores each character in memory unless CR, BS, or DEL.
*           - Backspace/Delete decrements Y.
*           - On CR, appends null (0x00) and returns.
*****************************************************************************/
IN_STRING:
Push r18
;r18 = 0 for count
IN_LOOP:
;call in_char to read a character
;check if character equals CR
        ;if it does store a null character and return
;check if char  = bs or DEL
        ;if it does dec the Y index
;else store the char at Y and inc Y
 ; Check if buffer full
; ignore extra input

Rjmp IN_LOOP
Pop r18
ret
```

```
/*****************************************************************************
* Name:     OUT_DATA_STRING
* Purpose:  output a character string stored in program memory
* Inputs:   none
* Outputs:  None
* Affected:
* Notes:
*****************************************************************************/
OUT_DATA_STRING:
OUT_LOOP:
;push registers
;read value pointed to by Y and increment Y
;if the character is null jump to return
        ;else move its value into r16 and jmp to OUT_LOOP
Rjmp OUT_STRING

OUT_DONE:
;pop registers
Ret
```

Lab5_7.asm

```
*/
.include "Atxmega128A1Udef.inc"
```

```
.equ CR = 13
.equ LF = 10
.equ String = 0x2000

.dseg

.cseg
.org 0x2000
String = “Arion Stern”

.org 0x0000
        rjmp MAIN

.org USARTD0_RXC_vect
Rjmp USART_ISR

.org 0x200
MAIN:
;init stack

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

; Initialize BLUE_PWM LED on PORTD pin 6

;enable global interrupts

REPEAT:
```

```
;toggle BLUE_LED

rjmp REPEAT



;*********************************************
/* INIT_USART_PINS
init portD pin 3 for output(PORTD0 TX pin) and
                pin 2 for input (PortD0 Rx pin)
subroutine: INIT_USART_PINS
function:       must set PORTD_PIN3 as output for TX pin
                        of USARTD0 and inital output value to 1
INPUT: none
OUTPUT: none
Destroys: r16
regs used: PORTD_DIR, PORTD_OUT
calls: none
*/
INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0

;set PORTD_PIN2 as input for rx of USARTd0

ret
```

```
;*********************************************
/* INIT_USART
init uart 0 on portd (PORTD0)
subroutine: INIT_USART
function:       init the usartd0's tx and rx
                        67000 Hz, 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:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale



;enable Tx and Rx

;enable interrupts in CTRLA
```

```
;set PMIC int level


ret


;*********************************************
/* OUT_CHAR receives a character via r16 and will
        poll the DREIF (data register empty flag) until its true;
subroutine: OUT_CHAR
function:       outputs the character in register r16 to the SCI Tx pin
                        after checking if the DREIF is empty.
                        The PC terminal program will take this
                        recieved 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

TX_POLL:
;load status register



;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
```

```
;send the character out over the USART

pop r17

ret

/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  output a character string stored in program memory
* Inputs:   none
* Outputs:  None
* Affected:
* Notes:
*****************************************************************************/
OUT_STRING:
;push registers
;read value pointed to by Z and increment z
;if the character is null jump to return
        ;else move its value into r16 and call OUT_CHAR
Rjmp OUT_STRING

RETURN:
;pop registers
Ret
;*********************************************
/* IN_CHAR polls the receive complete flag and will
                pass the receivedchar back to the calling routine in r16
subroutine: IN_CHAR
function:       receives typed char (sent by PC terminal
                        program through the PC to the PORTD0 USART Rx pin)
```

into register r16.

INPUT: none

OUTPUT: r16 = input from SCI

Destroys: r16 (result is transferred in this register)

regs used: USARTD0_STATUS, USARTD0_DATA

calls: none

**********/

IN_CHAR:

RX_POLL:

;load status register

;check if the receive flag (RXCIF) is set (to read in a character)

;read the character into r16

Ret

/**********

USART_ISR

**********/

USART_ISR:

;preserve status register

;clear the flag

;read a character (without IN_CHAR) read directly from data register since we know we received a byte

TX_WAIT:

;output that character (without OUT_CHAR)

```
;recover the status register

Reti
```

PROGRAM CODE

SECTION 2

Lab5_2.asm

```
/*
 * lab5_2.asm
 *
 *  Created: 10/17/2025 10:56:01 PM
 *   Author: arist
 */
 /*
 * lab5_2.asm
 *
 *  Created: 10/17/2025
 *   Author: Arion Stern
 */
; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configure USARTD0 for asynchronous serial transmission at 67 000 bps,
;              8 data bits, 1 stop bit, odd parity.  Continuously transmits the ASCII
;              character ‘U’ through Putty via the µPAD USB connection.

.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10
.dseg
.org 0x2000

.cseg
.org 0x0000
       rjmp MAIN

.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

REPEAT:

;load ‘U’
ldi r16, 'U'
rcall OUT_CHAR
;OUT_CHAR expects the argument in r16
```

```
Rjmp REPEAT



/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port D pins for USARTD0 operation (TX = PD3, RX = PD2).
* Inputs:   None
* Outputs:  None
* Affected: PORTD_DIR, PORTD_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0
ldi r16, BIT3
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16

;set PORTD_PIN2 as input for rx of USARTd0
ldi r16, BIT2
sts PORTD_DIRCLR, r16

ret


/*****************************************************************************
* Name:     INIT_USART
* Purpose:  Initialize USARTD0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_BAUDCTRLA/B, USARTD0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
*****************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
      ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                  USART_PMODE_ODD_gc | \
                  USART_CHSIZE_8BIT_gc)
      sts USARTD0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
      ldi r16, low(BSEL)
      sts USARTD0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
```

```
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTD0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( (1<<4) | (1<<3) )
sts USARTD0_CTRLB, r16

ret

/*****************************************************************************
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTD0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTD0_STATUS, USARTD0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTD0_DATA. This subroutine does not use interrupts.
*****************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTD0_STATUS


;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL

;send the character out over the USART
sts USARTD0_DATA, r16

pop r17

ret
```

Lab5_3.asm

```
/*
 * lab5_3.asm
 *
 *  Created: 10/17/2025 11:37:13 PM
 *   Author: arist
 */

; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configures USARTC0 for asynchronous serial transmission at 67 000 bps,
;              8 data bits, 1 stop bit, and odd parity. Continuously transmits the ASCII
;              character 'U' for measurement with the DAD to verify baud rate and frame width.
.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10

.dseg
.org 0x2000


.cseg
.org 0x0000
       rjmp MAIN

.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

REPEAT:

;load ‘U’
ldi r16, 'U'
rcall OUT_CHAR
;OUT_CHAR expects the argument in r16

Rjmp REPEAT



/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port C pins for USARTC0 operation (TX = PC3, RX = PC2).
* Inputs:   None
* Outputs:  None
* Affected: PORTC_DIR, PORTC_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
```

```
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portC0 tx is in bit 3
.equ BIT2 = 1<<2; portC0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTC_PIN3 as output for tx pin of USARTC0
ldi r16, BIT3
sts PORTC_OUTSET, r16
sts PORTC_DIRSET, r16

;set PORTC_PIN2 as input for rx of USARTC0
ldi r16, BIT2
sts PORTC_DIRCLR, r16

ret


/*****************************************************************************
* Name:     INIT_USART
* Purpose:  Initialize USARTC0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
* Affected: USARTC0_BAUDCTRLA/B, USARTC0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
*****************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
      ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                  USART_PMODE_ODD_gc | \
                  USART_CHSIZE_8BIT_gc)
      sts USARTC0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
      ldi r16, low(BSEL)
      sts USARTC0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTC0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( (1<<4) | (1<<3) )
sts USARTC0_CTRLB, r16

ret

/*****************************************************************************
```

```
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTC0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTC0_STATUS, USARTC0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTC0_DATA. This subroutine does not use interrupts.
*******************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTC0_STATUS


;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL

;send the character out over the USART
sts USARTC0_DATA, r16

pop r17

ret
```

Lab5_4.asm

```
/*
 * lab5_4.asm
 *
 *  Created: 10/18/2025 1:00:32 AM
 *   Author: arist
 */

; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configures USARTD0 for asynchronous serial transmission at 67 000 bps,
;              8 data bits, 1 stop bit, and odd parity. Outputs a null-terminated ASCII
;              string stored in program memory through PuTTY via the µPAD USB connection.


.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10


.dseg
.org 0x2000

.cseg
.org 0x2000

STRING:
    .db "Arion Stern",0


.org 0x0000
       rjmp MAIN

.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init Z pointer to point to STRING
ldi ZL, low(STRING<<1)
ldi ZH, high(STRING<<1)

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

rcall OUT_STRING

;carrage return and line feed
ldi r16, CR
rcall OUT_CHAR
ldi r16, LF
```

```
rcall OUT_CHAR

REPEAT:

Rjmp REPEAT



/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port D pins for USARTD0 operation (TX = PD3, RX = PD2).
* Inputs:   None
* Outputs:  None
* Affected: PORTD_DIR, PORTD_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0
ldi r16, BIT3
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16

;set PORTD_PIN2 as input for rx of USARTd0
ldi r16, BIT2
sts PORTD_DIRCLR, r16

ret


/*****************************************************************************
* Name:     INIT_USART
* Purpose:  Initialize USARTD0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_BAUDCTRLA/B, USARTD0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
*****************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
       ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                     USART_PMODE_ODD_gc | \
                     USART_CHSIZE_8BIT_gc)
       sts USARTD0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
       ldi r16, low(BSEL)
```

```
       sts USARTD0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTD0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( (1<<4) | (1<<3) )
sts USARTD0_CTRLB, r16

ret

/*****************************************************************************
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTD0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTD0_STATUS, USARTD0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTD0_DATA. This subroutine does not use interrupts.
*****************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTD0_STATUS


;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL

;send the character out over the USART
sts USARTD0_DATA, r16

pop r17

ret

/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  Transmit a null-terminated string stored in program memory through USARTD0.
* Inputs:   Z – points to start of the string in program memory.
* Outputs:  None
* Affected: Z-pointer, r16
* Notes:    Reads each character from program memory using LPM, stops at null (0x00),
*           and calls OUT_CHAR for each character.
*****************************************************************************/

OUT_STRING:
;push registers
NEXTCHAR:

;read value pointed to by Z and increment z
lpm r16, Z+
```

```
tst r16
;if the character is null jump to return
breq ENDSTRING
       ;else move its value into r16 and call OUT_CHAR
rcall OUT_CHAR
Rjmp NEXTCHAR

ENDSTRING:
;pop registers
ret
```

Lab5_5.asm

```
/*
 * lab5_5.asm
 *
 *  Created: 10/18/2025 1:43:07 AM
 *   Author: arist
 */
; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configures USARTD0 for asynchronous serial communication at 67 000 bps,
;              8 data bits, 1 stop bit, and odd parity. Implements character input and
;              echo functionality by reading a character from PuTTY and transmitting it
;              back through the µPAD USB interface.



.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10


.dseg
.org 0x2000

.cseg
;a.org 0x2000

;STRING:
 ;   .db "Arion Stern",0


.org 0x0000
       rjmp MAIN

.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init Z pointer to point to STRING
;ldi ZL, low(STRING<<1)
;ldi ZH, high(STRING<<1)

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART

REPEAT:

;read a character
rcall IN_CHAR
;output that character
```

```
rcall OUT_CHAR

;Repeat
rjmp REPEAT




/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port D pins for USARTD0 operation (TX = PD3, RX = PD2).
* Inputs:   None
* Outputs:  None
* Affected: PORTD_DIR, PORTD_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0
ldi r16, BIT3
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16

;set PORTD_PIN2 as input for rx of USARTd0
ldi r16, BIT2
sts PORTD_DIRCLR, r16

ret


/*****************************************************************************
* Name:     INIT_USART
* Purpose:  Initialize USARTD0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_BAUDCTRLA/B, USARTD0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
*****************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
       ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                     USART_PMODE_ODD_gc | \
                     USART_CHSIZE_8BIT_gc)
       sts USARTD0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
       ldi r16, low(BSEL)
```

```
       sts USARTD0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTD0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( (1<<4) | (1<<3) )
sts USARTD0_CTRLB, r16

ret

/*****************************************************************************
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTD0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTD0_STATUS, USARTD0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTD0_DATA. This subroutine does not use interrupts.
*****************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTD0_STATUS


;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL

;send the character out over the USART
sts USARTD0_DATA, r16

pop r17

ret

/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  Transmit a null-terminated string from program memory through USARTD0.
* Inputs:   Z – pointer to start of the string in program memory.
* Outputs:  None
* Affected: Z-pointer, r16
* Notes:    Reads each character from program memory using LPM, stops at null (0x00),
*           and calls OUT_CHAR for each character.
*****************************************************************************/

OUT_STRING:
;push registers
NEXTCHAR:

;read value pointed to by Z and increment z
lpm r16, Z+
```

```
tst r16
;if the character is null jump to return
breq ENDSTRING
       ;else move its value into r16 and call OUT_CHAR
rcall OUT_CHAR
Rjmp NEXTCHAR

ENDSTRING:
;pop registers
ret

/*****************************************************************************
* Name:     IN_CHAR
* Purpose:  Receive a single character through USARTD0.
* Inputs:   None
* Outputs:  r16 - received character (byte)
* Affected: USARTD0_STATUS, USARTD0_DATA
* Notes:    Polls the Receive Complete Flag (RXCIF) until data is available,
*           then reads from USARTD0_DATA. Does not use interrupts.
*****************************************************************************/
IN_CHAR:

RX_POLL:
;load status register
lds r16, USARTD0_STATUS

;check if the receive flag (RXCIF) is set (to read in a character)
sbrs r16, USART_RXCIF_bp
rjmp RX_POLL

;read the character into r16
lds r16, USARTD0_DATA

ret
```

Lab5_6.asm

```
/*
 * lab5_6.asm
 *
 *  Created: 10/18/2025 2:22:15 AM
 *   Author: arist
 */

; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configures USARTD0 for asynchronous serial communication at 67 000 bps,
;              8 data bits, 1 stop bit, and odd parity. Implements character string input
;              and output using data memory. The program receives a string from PuTTY,
;              supports backspace/delete correction, and echoes the completed string back
;              to the terminal after carriage return and line feed.



.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10
.equ BS = 0x08
.equ DEL = 0x7F
.equ StringSize = 32



.dseg
.org 0x2000
INPUT_TAB:
.byte StringSize

.cseg
;a.org 0x2000

;STRING:
 ;   .db "Arion Stern",0


.org 0x0000
       rjmp MAIN

.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART
```

```
REPEAT:

;init Y pointer to point to INPUT_TAB
ldi YL, low(INPUT_TAB)
ldi YH, high(INPUT_TAB)

;read the string from uart
rcall IN_STRING

;cr and lf
ldi r16, CR
rcall OUT_CHAR
ldi r16, LF
rcall OUT_CHAR

;reset the Y pointer and echo
ldi YL, low(INPUT_TAB)
ldi YH, high(INPUT_TAB)
rcall OUT_DATA_STRING

;cr and lf
ldi r16, CR
rcall OUT_CHAR
ldi r16, LF
rcall OUT_CHAR


rjmp REPEAT



/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port D pins for USARTD0 operation (TX = PD3, RX = PD2).
* Inputs:   None
* Outputs:  None
* Affected: PORTD_DIR, PORTD_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0
ldi r16, BIT3
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16

;set PORTD_PIN2 as input for rx of USARTd0
ldi r16, BIT2
sts PORTD_DIRCLR, r16

ret


/*****************************************************************************
```

```
* Name:     INIT_USART
* Purpose:  Initialize USARTD0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_BAUDCTRLA/B, USARTD0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
*****************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
       ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                     USART_PMODE_ODD_gc | \
                     USART_CHSIZE_8BIT_gc)
       sts USARTD0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
       ldi r16, low(BSEL)
       sts USARTD0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTD0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( (1<<4) | (1<<3) )
sts USARTD0_CTRLB, r16

ret

/*****************************************************************************
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTD0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTD0_STATUS, USARTD0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTD0_DATA. This subroutine does not use interrupts.
*****************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTD0_STATUS


;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL
```

```
;send the character out over the USART
sts USARTD0_DATA, r16

pop r17

ret

/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  Transmit a null-terminated string from program memory through USARTD0.
* Inputs:   Z – pointer to start of the string in program memory.
* Outputs:  None
* Affected: Z-pointer, r16
* Notes:    Reads each character from program memory using LPM, stops at null (0x00),
*           and calls OUT_CHAR for each character.
*****************************************************************************/

OUT_STRING:
;push registers
NEXTCHAR:

;read value pointed to by Z and increment z
lpm r16, Z+
tst r16
;if the character is null jump to return
breq ENDSTRING
       ;else move its value into r16 and call OUT_CHAR
rcall OUT_CHAR
Rjmp NEXTCHAR

ENDSTRING:
;pop registers
ret

/*****************************************************************************
* Name:     IN_CHAR
* Purpose:  Receive a single character through USARTD0.
* Inputs:   None
* Outputs:  r16 – received character (byte)
* Affected: USARTD0_STATUS, USARTD0_DATA
* Notes:    Polls the Receive Complete Flag (RXCIF) until data is available,
*           then reads from USARTD0_DATA. Does not use interrupts.
*****************************************************************************/

IN_CHAR:

RX_POLL:
;load status register
lds r16, USARTD0_STATUS

;check if the receive flag (RXCIF) is set (to read in a character)
sbrs r16, USART_RXCIF_bp
rjmp RX_POLL

;read the character into r16
lds r16, USARTD0_DATA

ret
```

```
/*****************************************************************************
* Name:     IN_STRING
* Purpose:  Receive a string from USARTD0 into data memory using the Y-pointer.
* Inputs:   Y – start address of buffer
* Outputs:  Stores each received character sequentially in data memory.
* Affected: Y-pointer, r16, r18
* Notes:    - Stores each character until carriage return is received.
*           - Handles backspace and delete by moving the Y-pointer back one byte.
*           - On CR, appends a null terminator and returns.
*           - Echoes valid characters back to the terminal as they are received.
*****************************************************************************/
IN_STRING:
Push r18
;r18 = 0 for count
ldi r18, 0
IN_LOOP:
;call in_char to read a character
rcall IN_CHAR
;check if character equals CR
cpi r16, CR
       ;if it does store a null character and return
       breq END_INPUT
;check if char  = bs or DEL
cpi r16, BS
       ;if it does dec the Y index
       breq BACKSPACE
cpi r16, DEL
       breq BACKSPACE
;else store the char at Y and inc Y
 ; Check if buffer full
 cpi r18, StringSize
 breq IN_LOOP ; ignore extra input

 ;store char
 st Y+, r16
 inc r18
 rcall OUT_CHAR
 rjmp IN_LOOP


BACKSPACE:
tst r18
breq IN_LOOP; dont go below data space
dec r18
sbiw YL, 1
;could also write LD r19, -Y and push and pop r19
;not required but wanted to actually erase:
       ldi r16, BS      ; backspace
       rcall OUT_CHAR
       ldi r16, ' '     ; overwrite
       rcall OUT_CHAR
       ldi r16, BS
       rcall OUT_CHAR



rjmp IN_LOOP


END_INPUT:
```

```
ldi r16, 0x00
st Y, r16

Pop r18
ret

/*****************************************************************************
* Name:     OUT_DATA_STRING
* Purpose:  Output a null-terminated string stored in data memory through USARTD0.
* Inputs:   Y – pointer to start of the string in data memory.
* Outputs:  None
* Affected: Y-pointer, r16
* Notes:    Reads each character sequentially from data memory until null (0x00),
*           and calls OUT_CHAR for each one.
*****************************************************************************/
OUT_DATA_STRING:
OUT_LOOP:
;push registers
;read value pointed to by Y and increment Y
ld r16, Y+
;if the character is null jump to return
tst r16
breq OUT_DONE
       ;else move its value into r16 and jmp to OUT_LOOP
       rcall OUT_CHAR
Rjmp OUT_LOOP

OUT_DONE:
;pop registers
Ret
```

Lab5_7.asm

```
/*
 * lab5_7.asm
 *
 *  Created: 10/18/2025 3:11:47 AM
 *   Author: arist
 */
 ; Lab 5, Section 22
; Name: Arion Stern
; Class #: 11303
; PI Name: Ian Santamauro
; Description: Configures USARTD0 for interrupt-based asynchronous serial communication
;              at 67 000 bps, 8 data bits, 1 stop bit, and odd parity. Implements an
;              interrupt-driven echo program that transmits back any received character
;              through PuTTY. Simultaneously toggles the BLUE_PWM LED on PD6 in the main
;              loop to demonstrate that USART reception operates independently via the
;              Receive Complete interrupt.


.include "Atxmega128A1Udef.inc"

.equ CR = 13
.equ LF = 10


.dseg
.org 0x2000

.cseg
;a.org 0x2000

;STRING:
 ;   .db "Arion Stern",0


.org 0x0000
       rjmp MAIN
.org USARTD0_RXC_vect
Rjmp USART_ISR



.org 0x200
MAIN:
;init stack
ldi r16, low(0x3FFF)
out CPU_SPL, r16
ldi r16, high(0x3FFF)
out CPU_SPH, r16

;init Z pointer to point to STRING
;ldi ZL, low(STRING<<1)
;ldi ZH, high(STRING<<1)

;init USART PINS
rcall INIT_USART_PINS
;init USART
rcall  INIT_USART
```

```
; Initialize BLUE_PWM LED on PORTD pin 6
ldi r18, (1<<6)
sts PORTD_OUTCLR, r18
sts PORTD_DIRSET, r18

sts PORTC_OUTSET, r18
sts PORTC_DIRSET, r18


; enable PMIC low-level interrupts and global
 ldi r18, PMIC_LOLVLEN_bm
 sts PMIC_CTRL, r18
 sei



REPEAT:
; Initialize BLUE_PWM LED on PORTD pin 6
ldi r18, (1<<6)
sts PORTD_OUTTGL, r18


rjmp REPEAT




/*****************************************************************************
* Name:     INIT_USART_PINS
* Purpose:  Configure Port D pins for USARTD0 operation (TX = PD3, RX = PD2).
* Inputs:   None
* Outputs:  None
* Affected: PORTD_DIR, PORTD_OUT
* Notes:    Sets TX pin high (idle state) and configures it as output.
*           Configures RX pin as input for incoming serial data.
*****************************************************************************/


INIT_USART_PINS:
.equ BIT3 = 1<<3; portd0 tx is in bit 3
.equ BIT2 = 1<<2; portd0 Rx is in bit 2

;set the tx line to default 1 (idle) as described in doc
;set PORTD_PIN3 as output for tx pin of USARTD0
ldi r16, BIT3
sts PORTD_OUTSET, r16
sts PORTD_DIRSET, r16

;set PORTD_PIN2 as input for rx of USARTd0
ldi r16, BIT2
sts PORTD_DIRCLR, r16

ret


/*****************************************************************************
* Name:     INIT_USART
* Purpose:  Initialize USARTD0 for asynchronous serial communication at 67 000 bps,
*           8 data bits, 1 stop bit, and odd parity.
* Inputs:   None
* Outputs:  None
```

```
* Affected: USARTD0_BAUDCTRLA/B, USARTD0_CTRLB/C
* Notes:    - fPER = 2 MHz, CLK2X = 0 (normal speed).
*           - Enables both transmitter (TXEN) and receiver (RXEN).
********************************************************************************/

INIT_USART:
;equates
.equ BSel = 14
.equ BScale = -4 ; 67000 Hz
;push registers
push r16

; set parity to odd, 8 bit frame, 1 stop bit (0x3)
       ldi r16, (USART_CMODE_ASYNCHRONOUS_gc | \
                     USART_PMODE_ODD_gc | \
                     USART_CHSIZE_8BIT_gc)
       sts USARTD0_CTRLC, r16

;initialize baud rate
;set BAUDCTRLA with only the lower 8 bits of Bsel
       ldi r16, low(BSEL)
       sts USARTD0_BAUDCTRLA, r16

;set baudctrlb to BSCALE | BSEL lower 4 bits are upper 4 bits
; of BSel and upper 4 bits are the BScale

ldi r16, ( (BSCALE<<4) | high(BSEL) )
sts USARTD0_BAUDCTRLB, r16

;enable Tx and Rx
ldi r16, ( USART_TXEN_bm) | (USART_RXEN_bm)
sts USARTD0_CTRLB, r16

;enable interrupts in CTRLA
ldi r16, USART_RXCINTLVL_LO_gc
 sts USARTD0_CTRLA, r16

 ;pop registers
 pop r16

ret

/********************************************************************************
* Name:     OUT_CHAR
* Purpose:  Transmit a single character stored in r16 through USARTD0.
* Inputs:   r16 – character (byte) to send
* Outputs:  None
* Affected: USARTD0_STATUS, USARTD0_DATA, r17
* Notes:    Polls the Data Register Empty Flag (DREIF) until ready, then writes r16
*           to USARTD0_DATA. This subroutine does not use interrupts.
********************************************************************************/


OUT_CHAR:
push r17

TX_POLL:
;load status register
lds r17, USARTD0_STATUS
```

```
;check if the data register empty flag (DREIF) is set (to send out a char)
;else go back to polling
 sbrs r17, USART_DREIF_bp ; 5
rjmp TX_POLL

;send the character out over the USART
sts USARTD0_DATA, r16

pop r17

ret

/*****************************************************************************
* Name:     OUT_STRING
* Purpose:  Transmit a null-terminated string from program memory through USARTD0.
* Inputs:   Z - pointer to start of string in program memory.
* Outputs:  None
* Affected: Z-pointer, r16
* Notes:    Reads each character from program memory using LPM, stops at null (0x00),
*           and calls OUT_CHAR for each character.
*****************************************************************************/

OUT_STRING:
;push registers
NEXTCHAR:

;read value pointed to by Z and increment z
lpm r16, Z+
tst r16
;if the character is null jump to return
breq ENDSTRING
       ;else move its value into r16 and call OUT_CHAR
rcall OUT_CHAR
Rjmp NEXTCHAR

ENDSTRING:
;pop registers
ret

/*****************************************************************************
* Name:     IN_CHAR
* Purpose:  Receive a single character through USARTD0 (polling method).
* Inputs:   None
* Outputs:  r16 - received character (byte)
* Affected: USARTD0_STATUS, USARTD0_DATA
* Notes:    Polls the Receive Complete Flag (RXCIF) until data is available,
*           then reads from USARTD0_DATA. Does not use interrupts.
*****************************************************************************/

IN_CHAR:

RX_POLL:
;load status register
lds r16, USARTD0_STATUS

;check if the receive flag (RXCIF) is set (to read in a character)
sbrs r16, USART_RXCIF_bp
rjmp RX_POLL

;read the character into r16
lds r16, USARTD0_DATA
```

```
ret



/*****************************************************************************
* Name:     USART_ISR
* Purpose:  Interrupt Service Routine for USARTD0 Receive Complete interrupt.
* Inputs:   None
* Outputs:  None
* Affected: USARTD0_DATA, PORTC_OUT, CPU_SREG
* Notes:    - Executes automatically when a character is received via USARTD0.
*           - Reads received byte from USARTD0_DATA (clearing RXCIF).
*           - Immediately echoes the received character back by writing it to
*             USARTD0_DATA.
*           - Toggles PORTC LED for debugging confirmation of ISR activity.
*****************************************************************************/

USART_ISR:
;preserve status register
push r16
lds r16, CPU_SREG
push r16

    ; toggle LED on ISR trigger for debugging
    ldi r16, (1<<6)
    sts PORTC_OUTTGL, r16
;clear the flag (READING FROM DATA CLEARS IT IN ISR OR WRITE 1 to it)
;read a character (without IN_CHAR) read directly from data register since we know we received a byte
lds r16, USARTD0_DATA

;output that character (without OUT_CHAR)
sts USARTD0_DATA, r16

;recover the status register
pop r16
sts CPU_SREG, r16
pop r16

Reti
```

APPENDIX

Supporting ASM/C Code and Additional Information

- Included/Referenced Files and Headers:
  - ATxmega128A1Udef.inc
  - Used code from the recorded lecture videos
- Additional Screenshots/Images:

Figure 1: Annotated waveforms of a single character transmission frame for 'U'

The ideal baud Rate is 67,000, and the calculated baud rate using a BSEL of 14 and BSCALE of -4 is 66666.66667. However, the measured baud rate is 67,461 which has a 0.69 percent error compared to the ideal baud rate which is acceptable for the purposes of our lab. The width of the single data bit is 14.823 microseconds.

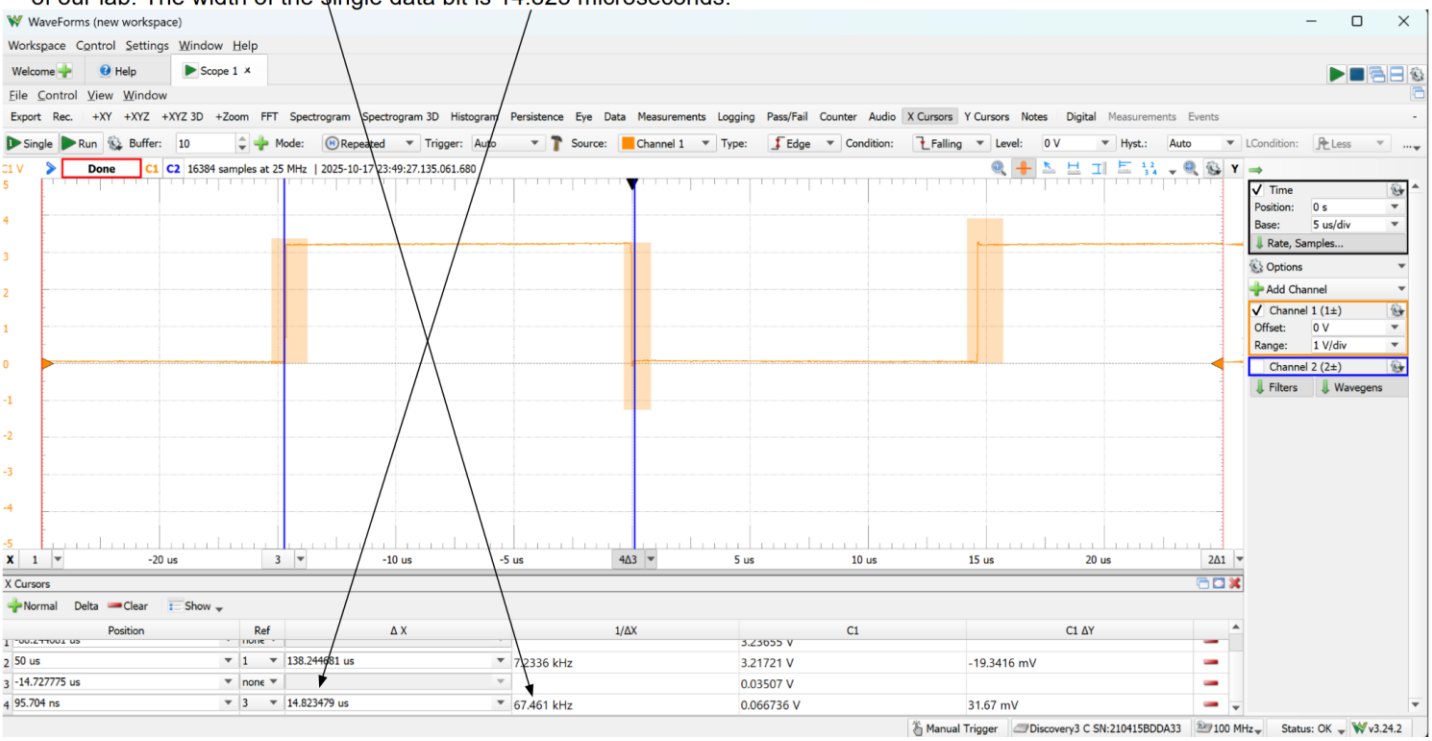


Figure 2: Annotated Waveform of a single data bit

Figure 3: Verifying that part 2 constantly outputs 'U'

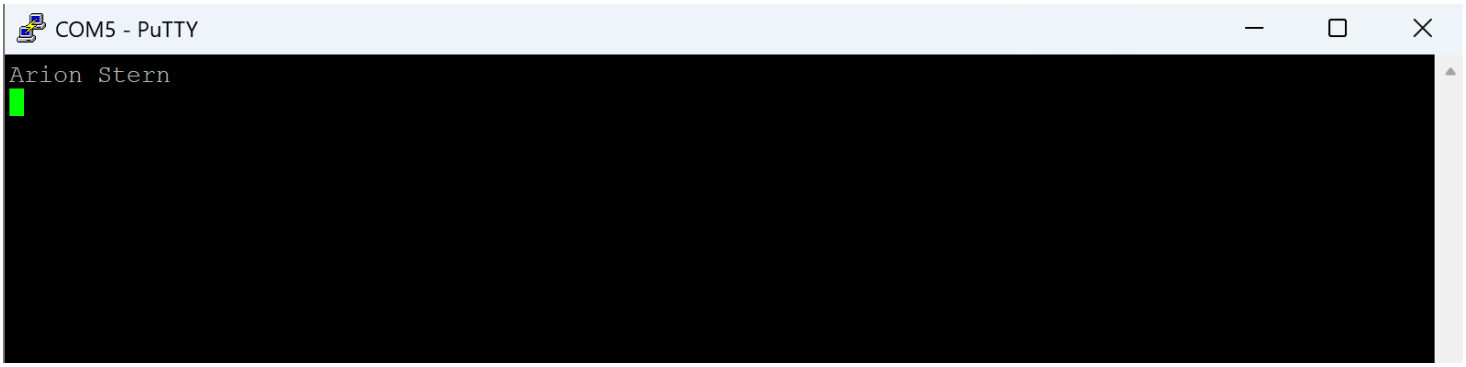


Figure 4: Verifying that part 4 (OUT_STRING) displays my name stored in program memory

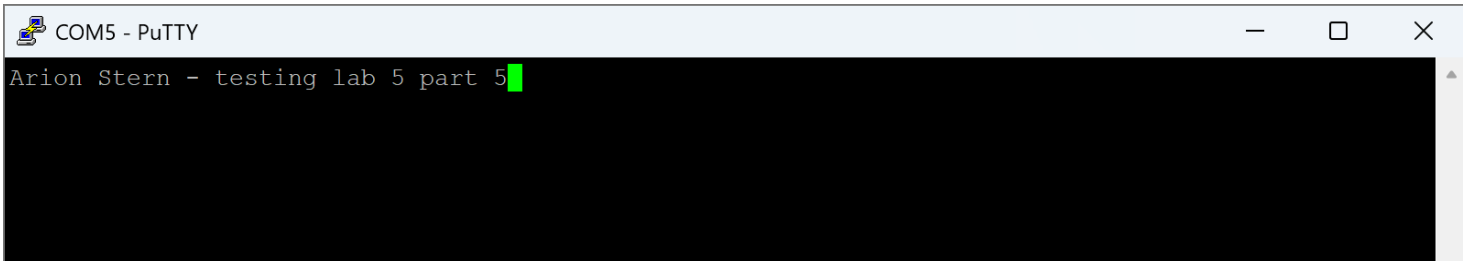


Figure 5: Verifying that part 5 echoes characters I type

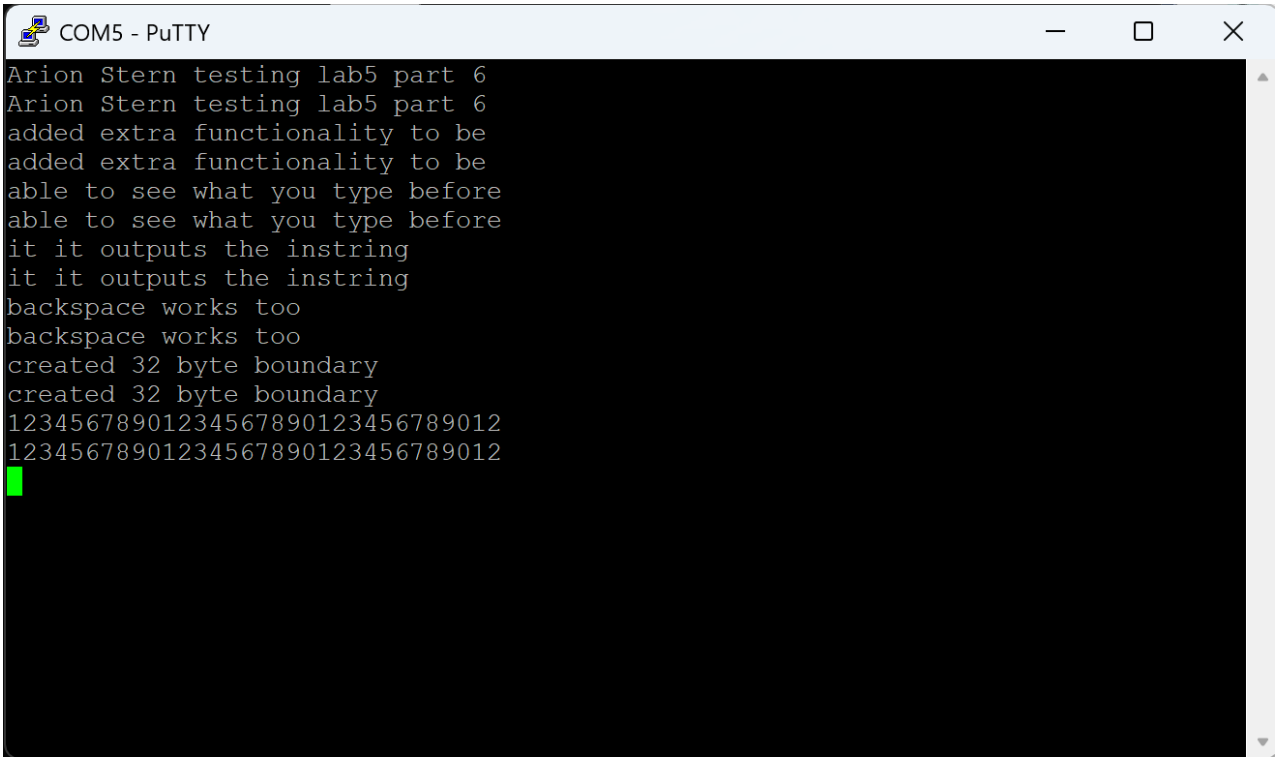


Figure 6: Verifying that part 6 works as intended. I added a feature to let the user see what they are typing into the IN_STRING, which will be echoed back after hitting enter. The backspace key works and I also implemented a 32 byte boundary.

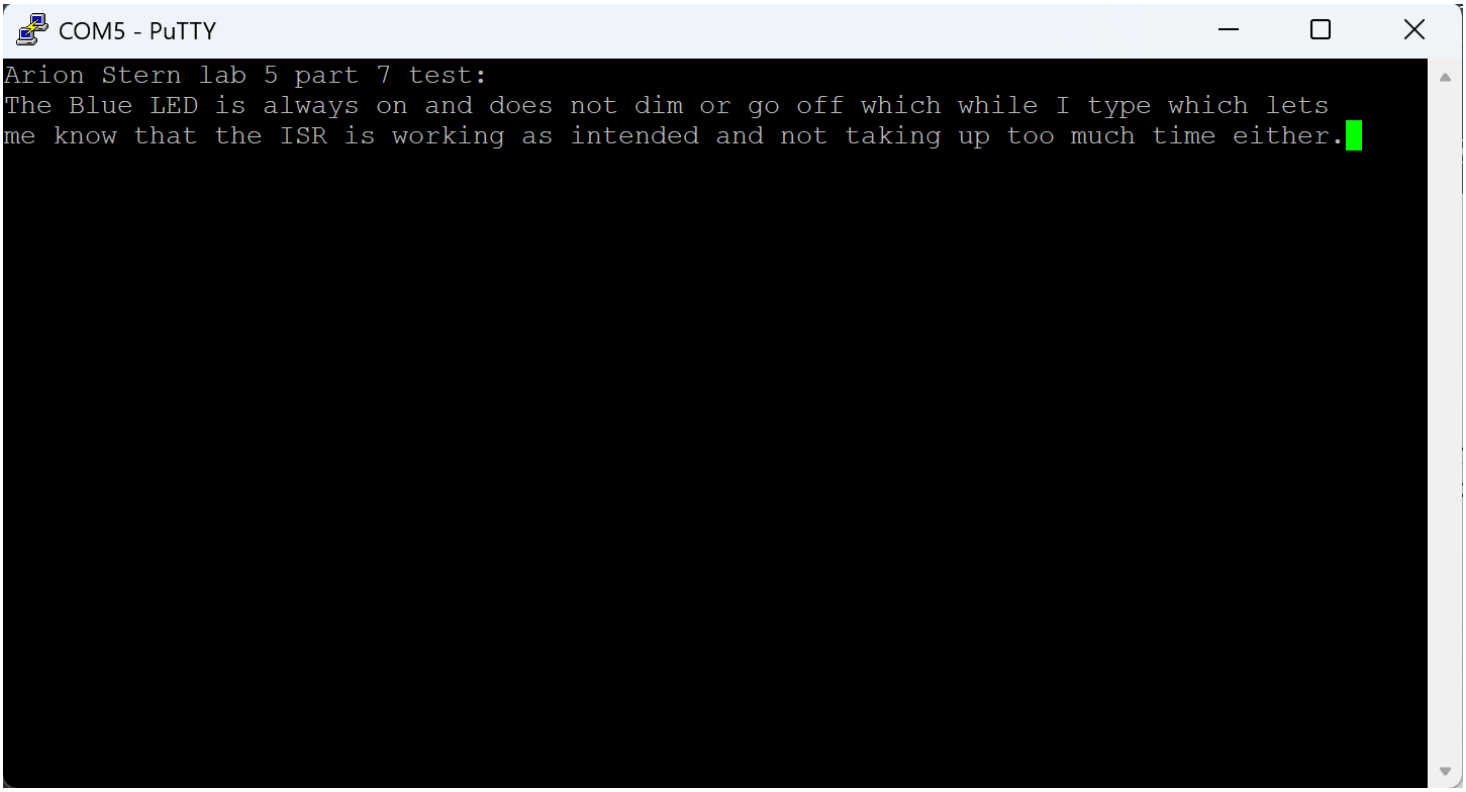


Figure 7: Verifying that my part 7 ISR works correctly and the blue LED stays on and does not flicker when I type