Lab 4
Jineidy Mak
Secton 7F31

Pre-Lab Questions:

1. List the XMEGA's USART registers used in your programs and briefly describe their functions.
USARTD0_CTRLA - Used to set up interrupt levels for the USART Rx and Tx pins.
USARTD0_CTRLB - Used to enable the Tx and Rx pins.
USARTD0_CTRLC - Used to set up the frame used to pass information.
USARTD0_BAUDCTRLA - Holds the lower 8 bits of BSel.
USARTD0_BAUDCTRLB - Holds the upper 4 bits of BSel and the BScale bits.

2. What is the difference between synchronous and asynchronous communication?
Synchronous communication relies on a third party clock while asynchronous communication relies on "handshakes".
3. What is the difference between serial and parallel communication?
Serial communication can only transmit data on one line this is why there are checks to see if information has been received/sent properly.
4. List the number of bounces from part A of this lab. How long (in ms) is your delay routine for debouncing?
Honestly there was little to no bounce in my switch it did take a while from the switch in the level. 2ms.
5. What is the maximum possible baud you can use for asynchronous communication if your board runs at 2 MHz? Support your answer with the values you would place in any special registers that are needed. The baud rate must be less than the peripheral clock frequency.
Fbaud <= fper/16 = 2MHz/16 = .125MHz. Which means BSel and BScale should both be zero.

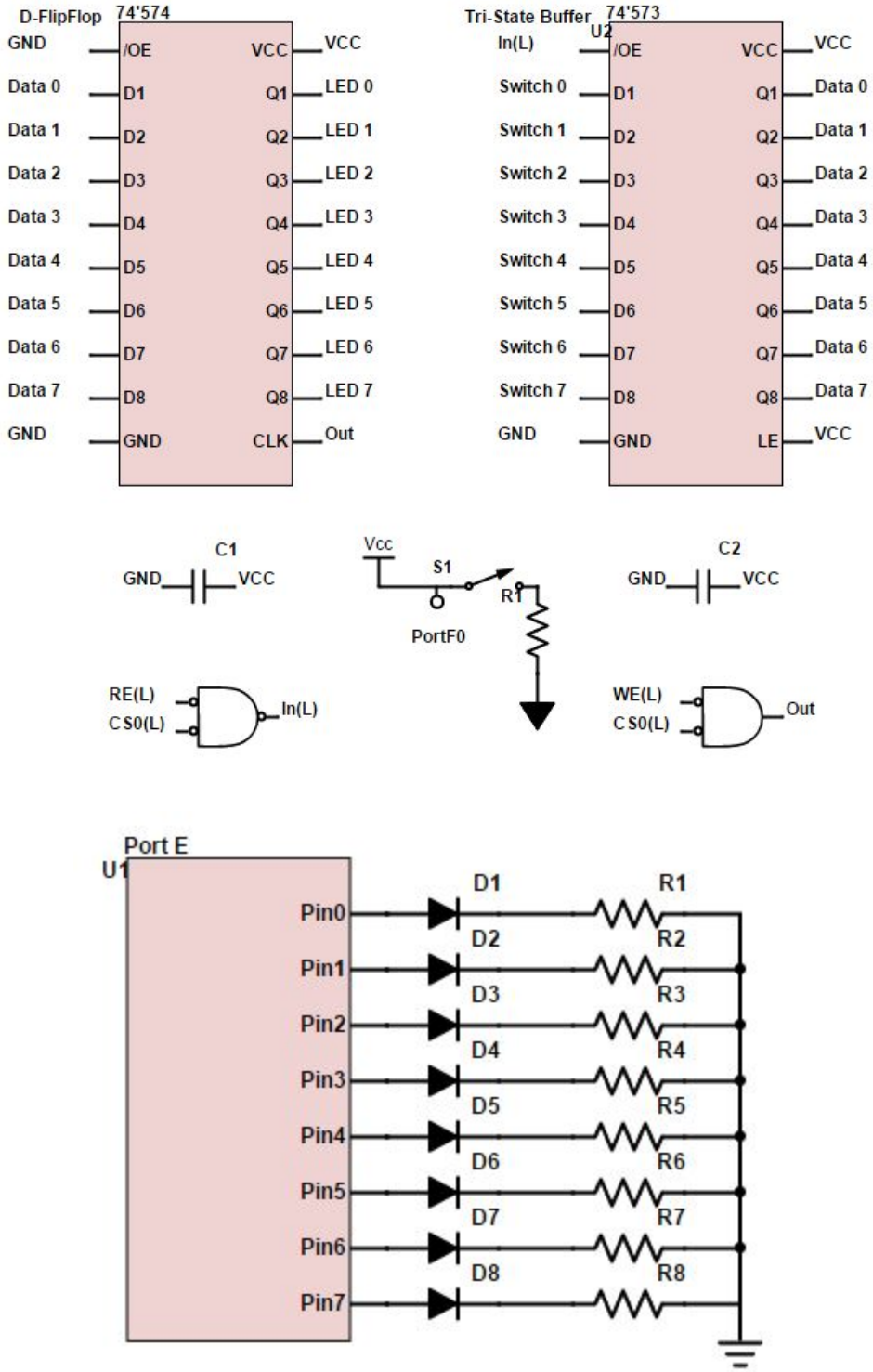
Problems Encountered:

/OE on D-Flip Flops should be disabled when not in use.

Future Work/Applications:

In embedded systems communication between different external peripherals can be done using the USART.

Schematics:


D-FlipFlop 74'574
GND
Data 0
Data 1
Data 2
Data 3
Data 4
Data 5
Data 6
Data 7
GND
/OE
D1
D2
D3
D4
D5
D6
D7
D8
GND
VCC
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
CLK
VCC
LED 0
LED 1
LED 2
LED 3
LED 4
LED 5
LED 6
LED 7
Out
Tri-State Buffer 74'573
U2
In(L)
Switch 0
Switch 1
Switch 2
Switch 3
Switch 4
Switch 5
Switch 6
Switch 7
GND
/OE
D1
D2
D3
D4
D5
D6
D7
D8
GND
VCC
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
LE
VCC
Data 0
Data 1
Data 2
Data 3
Data 4
Data 5
Data 6
Data 7
VCC
C1
GND
VCC
Vcc
S1
R1
PortF0
C2
GND
VCC
RE(L)
CS0(L)
In(L)
WE(L)
CS0(L)
Out
Port E
U1
Pin0
Pin1
Pin2
Pin3
Pin4
Pin5
Pin6
Pin7
D1
D2
D3
D4
D5
D6
D7
D8
R1
R2
R3
R4
R5
R6
R7
R8

Decoding Logic:

None for this lab. Some shown in previous section.

Pseudocode/Flowcharts:

PartA:

1. Initialize Stack.
2. Initialize Interrupts.
3. Create an endless loop and attempt to trigger interrupt.
4. Every time an interrupt is triggered and the pin is high record the count. The count should not reset so a singled register should be used for counter.

PartC:

1. Initialize Stack.
2. Initialize USART.
3. Rx Poll, check if the flag in status register is set, if so then insert the incoming data to a register. If not keep looping.
4. Tx Poll, check if the DREIF flag is set if so then send data in a register to outgoing data. If not keep looping.
5. For strings Keep calling the Rx or Tx poll as the pointer in the table keeps incrementing. If the table value is NUL then end the output/input.

PartD:

1. Same as part C, insert a 1000uS delay after the output of a single character this time.

PartE:

1. Same as Part C, the strings will be longer because of the Menu/Answers.
2. The input will take in a menu selection and then the assembly will use compare statements to check each one. The Z pointer will loaded to the answer that corresponds to the option. If no option is chosen then nothing will happen unless an option is chosen.

PartF:

1. Initialize Interrupt for USART.
2. Like PartC except whenever there is an input it will immediately be send to the output.
3. Instead of and endless loop like Part A we will be toggling an LED in the endless loop in Part F while we wait for an interrupt to occur.

Program Code:

PartA:

```
;
; lab4a.asm
;
; Created: 7/3/2016 7:22:06 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x3FFF
.equ Port_Start = 0x8000

.org 0x0000
		rjmp MAIN

.org PORTF_INT0_VECT
		jmp ISR_LED_COUNT

.org 0x100

MAIN:
				ldi YL, high(stack_init)
				out CPU_SPH, YL				;Initialize the high byte of
the stack pointer.
				ldi YL, low(stack_init)
				out CPU_SPL, YL				;Initialize the low byte of the
stack pointer.

				ldi R16, 0xFF
				sts PORTE_DIRSET, R16

				rcall INIT_INT				;Call the routine to intialize
the interrupt.
				nop
				ldi r17, 0x00					;Initialize the
interrupt count.

LOOP:							;Endless loop
while waiting for interrupt trigger.
				rjmp LOOP


INIT_INT:
				ldi R16, 0x01
				sts PORTF_INT0MASK, R16		;Interrupt to trigger from PIN 0 Port F.
				sts PORTF_OUT, R16				;Output Default '1' from PIN
0 Port F.
				sts PORTF_DIRCLR, R16		;Set PIN0 Port F to input.
				sts PORTF_INTCTRL, R16		;Set the interrupt level to low level.
				sts PORTF_PIN0CTRL, R16		;Set the interrupt to trigger on a
rising edge.
```

```
                        sts PMIC_CTRL, R16                      ;Turn on low-level
interrupts.

                        sei                                     ;Turn on
global interrupt flag.
                        ret


ISR_LED_COUNT:
                        push R19
                        lds R19, CPU_SREG                       ;Push the status register
onto the stack.
                        push R19
                        call DELAY_p5ms
                        nop
                        lds R18, PORTF_IN                       ;Read port E data.
                        SBRC R18, 0                             ;If bit 0 is cleared
skip the next instruction.
                        inc R17                                 ;Increment R16 if
the bit 0 was set.
                        sts PORTE_OUT, R17              ;Output to LEDs.
                        nop
                        sts PORTF_INTFLAGS, R16         ;Clear the interrupt flag.
                        pop R19
                        sts CPU_SREG, R19
                        pop R19
                        reti

DELAY_p5ms:             ;Delay assumes each instruction is .5uS long and also that the entire delay routine is 10
instructions long. (5uS)
                        ldi R21, 4                              ;.125ms x 4 = .5ms
DELAY_LOOP:     ldi R20, 50                             ;25 cycles. .25uS x 50  = .125ms
DELAY_p15ms:    dec R20
                        nop
                        nop
                        nop
                        brne DELAY_p15ms
                        dec R21
                        brne DELAY_LOOP
                        ret
```

Part C:

```
;
; lab4c.asm
;
; Created: 7/4/2016 3:54:55 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x3FFF
.equ bsel = 107
```

```
.equ bscale = -5

.dseg

Input_String: .byte 10
.cseg

.org 0x0000
        rjmp MAIN

.org 0x0100

String: .db "Hello World", 0x0D, 0x0A, 0x00

MAIN:
                ldi YL, high(stack_init)        ;Initialize the stack pointer.
                sts CPU_SPH, YL
                ldi YL, low(stack_init)
                sts CPU_SPL, YL
                rcall USART_INIT                ;Initialize the USART.

                ldi ZL, low(String << 1)        ;Initialize Z pointer to read data.
                ldi ZH, high(String << 1)
                ldi XL, low(Input_String)       ;Initialize X pointer to store data.
                ldi XH, high(Input_String)


                ;ldi R16, 'a'
START:          rcall IN_CHAR
                cpi R16, 'S'                    ;If character is 'S' then start
Output.
                brne START
                ;rcall CHAR_OUT
                rcall OUT_STRING
REPEAT:
                rjmp REPEAT                     ;Endless loop of outputting
the same table.



USART_INIT:
                ldi R16, 0x0A
                sts PORTQ_DIRSET, R16           ;Set Port Q pins 1 and 3 to output.
                sts     PORTQ_OUTCLR, R16       ;Output Defaul 0 to pins 1
and 3 of Port Q.
                ldi R16, 0x08                   ;Set Pin 3 of Port D to
output (Tx).
                sts PORTD_DIRSET, R16
                sts PORTD_OUTSET, R16           ;Set the default value as 1.
                ldi R16, 0x04
                sts PORTD_DIRCLR, R16           ;Set RX pin as input.
                ldi R16, 0x18
                sts USARTD0_CTRLB, R16          ;Turn on the transmitter (Tx) and
receiver (Rx)
                ldi R16, 0x03
                sts USARTD0_CTRLC, R16          ;Async Mode, No Parity, 1 Stop bit,
8-bit Frame.
```

```
                                        ldi R16, (bsel & 0xFF)          ;Lower 8-bits of the 12-bit bsel value
in register BAUDCTRLA
                                        sts USARTD0_BAUDCTRLA, R16
                                        ldi R16, ((bscale << 4) & 0xF0) | ((bsel >>  8) & 0x0F)
                                        sts USARTD0_BAUDCTRLB, R16      ;Most signification 4 bits of bsel
value in the least significant 4 bits of BAUDCTRLB.
                                        ret                                                     ;

CHAR_OUT:
                                        push R17
TX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 5                                     ;If the DREIF flag
(Bit 5) is set then send to out or else wait until set.
                                        rjmp TX_POLL
                                        sts USARTD0_DATA, R16           ;Send the character in R16 out.
                                        pop R17
                                        ret

IN_CHAR:

RX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 7                                     ;If the Rx Flag is
set then send to in or else wait until set.
                                        rjmp RX_POLL
                                        lds R16, USARTD0_DATA           ;Read the character into R16.
                                        ret




OUT_STRING:
                                        lpm R16, Z+                                     ;Load a value from
string table to R16.
                                        cpi R16, 0x00                           ;If the character is NULL
then end the output of the string.
                                        breq END_OUT
                                        rcall CHAR_OUT                  ;Output the next character
                                        rjmp OUT_STRING                 ;Repeat until NULL
character is reached.
END_OUT:                        ret


IN_STRING:
                                        rcall IN_CHAR                   ;Read the input
                                        cpi R16, 0x00                   ;If the character is null then
end the reading of the string.
                                        breq END_IN
                                        st X+, R16                                      ;Else store the
character in the location X is pointed to.
                                        rjmp IN_STRING                  ;Repeat until NULL
character is reached.
END_IN:                                 ret
```

PartD:

```
;
; lab4d.asm
;
; Created: 7/4/2016 3:54:55 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x3FFF
.equ bsel = 107
.equ bscale = -5

.dseg

Input_String: .byte 10
.cseg

.org 0x0000
		rjmp MAIN

.org 0x0100

String: .db "Hello World", 0x0D, 0x0A, 0x00

MAIN:
					ldi YL, high(stack_init)	;Initialize the stack pointer.
					sts CPU_SPH, YL
					ldi YL, low(stack_init)
					sts CPU_SPL, YL
					rcall USART_INIT			;Initialize the USART.

					ldi ZL, low(String << 1)	;Initialize Z pointer to read data.
					ldi ZH, high(String << 1)
					ldi XL, low(Input_String)	;Initialize X pointer to store data.
					ldi XH, high(Input_String)


REPEAT:				ldi R16, 'U'
					rcall CHAR_OUT
					rcall DELAY_1ms
					rjmp REPEAT					;Endless loop of outputting
the same table.



USART_INIT:
					ldi R16, 0x0A
					sts PORTQ_DIRSET, R16		;Set Port Q pins 1 and 3 to output.
					sts	PORTQ_OUTCLR, R16		;Output Defaul 0 to pins 1
and 3 of Port Q.
					ldi R16, 0x08				;Set Pin 3 of Port D to
output (Tx).
```

```
                                        sts PORTD_DIRSET, R16
                                        sts PORTD_OUTSET, R16           ;Set the default value as 1.
                                        ldi R16, 0x04
                                        sts PORTD_DIRCLR, R16           ;Set RX pin as input.
                                        ldi R16, 0x18
                                        sts USARTD0_CTRLB, R16          ;Turn on the transmitter (Tx) and
receiver (Rx)
                                        ldi R16, 0x03
                                        sts USARTD0_CTRLC, R16          ;Async Mode, No Parity, 1 Stop bit,
8-bit Frame.
                                        ldi R16, (bsel & 0xFF)          ;Lower 8-bits of the 12-bit bsel value
in register BAUDCTRLA
                                        sts USARTD0_BAUDCTRLA, R16
                                        ldi R16, ((bscale << 4) & 0xF0) | ((bsel >>  8) & 0x0F)
                                        sts USARTD0_BAUDCTRLB, R16      ;Most signification 4 bits of bsel
value in the least significant 4 bits of BAUDCTRLB.
                                        ret                                                     ;

CHAR_OUT:
                                        push R17
TX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 5                                     ;If the DREIF flag
(Bit 5) is set then send to out or else wait until set.
                                        rjmp TX_POLL
                                        sts USARTD0_DATA, R16           ;Send the character in R16 out.
                                        pop R17
                                        ret

IN_CHAR:

RX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 7                                     ;If the Rx Flag is
set then send to in or else wait until set.
                                        rjmp RX_POLL
                                        lds R16, USARTD0_DATA           ;Read the character into R16.
                                        ret




OUT_STRING:
                                        lpm R16, Z+                                     ;Load a value from
string table to R16.
                                        cpi R16, 0x00                           ;If the character is NULL
then end the output of the string.
                                        breq END_OUT
                                        rcall CHAR_OUT                  ;Output the next character
                                        rjmp OUT_STRING                 ;Repeat until NULL
character is reached.
END_OUT:                ret


IN_STRING:
                                        rcall IN_CHAR                           ;Read the input
```

```
                                        cpi R16, 0x00                   ;If the character is null then
end the reading of the string.
                                        breq END_IN
                                        st X+, R16                              ;Else store the
character in the location X is pointed to.
                                        rjmp IN_STRING                  ;Repeat until NULL
character is reached.
END_IN:                                 ret


DELAY_1ms:
                                        ldi R21, 8                              ;.125ms x 80 =
10ms
DELAY_LOOP:             ldi R20, 51                             ;25 cycles. .25uS x 50  =
.125ms
DELAY_p15ms:            dec R20
                                        nop
                                        nop
                                        nop
                                        brne DELAY_p15ms
                                        dec R21
                                        brne DELAY_LOOP
                                        ret
```

PartE:

```
;
; lab4e.asm
;
; Created: 7/4/2016 3:54:55 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x3FFF
.equ bsel = 107
.equ bscale = -5

.dseg

Input_String: .byte 10
.cseg

.org 0x0000
        rjmp MAIN

.org 0x0100

String: .db "James' Favorite :", 0x0D, 0x0A, "1. Book", 0x0D, 0x0A, "2. Actor,Actress,Reality Star", 0x0D, 0x0A, "3.
IceCream,Yogurt Flavor", 0x0D, 0x0A, "4. Food", 0x0D, 0x0A, "5. Pizza Topping", 0x0D, 0x0A, "6. Redisplay Menu", 0x0D,
0x0A, "ESC: Exit", 0x0D, 0x0A, 0x00
End_Words: .db "Goodbye!", 0x0D, 0x0A, 0x00

Ans1: .db "James' favorite book is, I don't read.", 0x0D, 0x0A, 0x00
```

```
Ans2: .db "James' favorite Actor/Actress/Reality Star, don't have one", 0x0D, 0x0A, 0x00
Ans3: .db "James' favorite IceCream/Yogurt Flavor is coconut!", 0x0D, 0x0A, 0x00
Ans4: .db "James' favorite Food is homecooked!", 0x0D, 0x0A, 0x00
Ans5: .db "James' favorite Pizza Topping is peperonis!, duh", 0x0D, 0x0A, 0x00

MAIN:
                    ldi YL, high(stack_init)    ;Initialize the stack pointer.
                    sts CPU_SPH, YL
                    ldi YL, low(stack_init)
                    sts CPU_SPL, YL
                    rcall USART_INIT            ;Initialize the USART.


                    ldi XL, low(Input_String)   ;Initialize X pointer to store data.
                    ldi XH, high(Input_String)



START:              rcall IN_CHAR
                    cpi R16, 'S'                ;If character is 'S' then start
Output.
                    brne START

MENU:               ldi ZL, low(String << 1)    ;Initialize Z pointer to read data.
                    ldi ZH, high(String << 1)
                    rcall OUT_STRING            ;Display the Menu.
REPEAT:             rcall IN_CHAR
                    cpi R16, 0x00               ;If there is nothing in the
Input the repeat. (Just so we don't get an input we don't want).
                    breq REPEAT
                    cpi R16, 0x31               ;Check to see if first choice
was chosen. Then the next, it will run down the menu.
                    brne CHOICE_2
                    ldi ZL, low(Ans1 << 1)
                    ldi ZH, high(Ans1 << 1)
                    rcall OUT_STRING
                    rjmp REPEAT                 ;Go back to the
top when an answer is output.
CHOICE_2:           cpi R16, 0x32
                    brne CHOICE_3
                    ldi ZL, low(Ans2 << 1)
                    ldi ZH, high(Ans2 << 1)
                    rcall OUT_STRING
                    rjmp REPEAT
CHOICE_3:           cpi R16, 0x33
                    brne CHOICE_4
                    ldi ZL, low(Ans3 << 1)
                    ldi ZH, high(Ans3 << 1)
                    rcall OUT_STRING
                    rjmp REPEAT
CHOICE_4:           cpi R16, 0x34
                    brne CHOICE_5
                    ldi ZL, low(Ans4 << 1)
                    ldi ZH, high(Ans4 << 1)
                    rcall OUT_STRING
                    rjmp REPEAT
CHOICE_5:           cpi R16, 0x35
```

```
                        brne CHOICE_6
                        ldi ZL, low(Ans5 << 1)
                        ldi ZH, high(Ans5 << 1)
                        rcall OUT_STRING
                        rjmp REPEAT
CHOICE_6:           cpi R16, 0x36
                        brne CHOICE_ESC
                        rjmp MENU
CHOICE_ESC:         cpi R16, 0x1B
                        brne REPEAT
                        ldi ZL, low(End_Words)
                        ldi ZH, high(End_Words)
                        rcall OUT_STRING

                        rjmp START

;Endless loop of outputting the same table.



USART_INIT:
                        ldi R16, 0x0A
                        sts PORTQ_DIRSET, R16       ;Set Port Q pins 1 and 3 to output.
                        sts     PORTQ_OUTCLR, R16           ;Output Defaul 0 to pins 1
and 3 of Port Q.
                        ldi R16, 0x08                       ;Set Pin 3 of Port D to
output (Tx).
                        sts PORTD_DIRSET, R16
                        sts PORTD_OUTSET, R16       ;Set the default value as 1.
                        ldi R16, 0x04
                        sts PORTD_DIRCLR, R16       ;Set RX pin as input.
                        ldi R16, 0x18
                        sts USARTD0_CTRLB, R16      ;Turn on the transmitter (Tx) and
receiver (Rx)
                        ldi R16, 0x03
                        sts USARTD0_CTRLC, R16      ;Async Mode, No Parity, 1 Stop bit,
8-bit Frame.
                        ldi R16, (bsel & 0xFF)      ;Lower 8-bits of the 12-bit bsel value
in register BAUDCTRLA
                        sts USARTD0_BAUDCTRLA, R16
                        ldi R16, ((bscale << 4) & 0xF0) | ((bsel >>  8) & 0x0F)
                        sts USARTD0_BAUDCTRLB, R16  ;Most signification 4 bits of bsel
value in the least significant 4 bits of BAUDCTRLB.
                        ret                                                 ;

CHAR_OUT:
                        push R17
TX_POLL:
                        lds R17, USARTD0_STATUS     ;Load the USART status register.
                        sbrs R17, 5                                 ;If the DREIF flag
(Bit 5) is set then send to out or else wait until set.
                        rjmp TX_POLL
                        sts USARTD0_DATA, R16       ;Send the character in R16 out.
                        pop R17
                        ret

IN_CHAR:
```

```
RX_POLL:
                                        lds R17, USARTD0_STATUS     ;Load the USART status register.
                                        sbrs R17, 7                 ;If the Rx Flag is
set then send to in or else wait until set.
                                        rjmp RX_POLL
                                        lds R16, USARTD0_DATA       ;Read the character into R16.
                                        ret




OUT_STRING:
                                        lpm R16, Z+                 ;Load a value from
string table to R16.
                                        cpi R16, 0x00               ;If the character is NULL
then end the output of the string.
                                        breq END_OUT
                                        rcall CHAR_OUT              ;Output the next character
                                        rjmp OUT_STRING             ;Repeat until NULL
character is reached.
END_OUT:                ret


IN_STRING:
                                        rcall IN_CHAR               ;Read the input
                                        cpi R16, 0x00               ;If the character is null then
end the reading of the string.
                                        breq END_IN
                                        st X+, R16                  ;Else store the
character in the location X is pointed to.
                                        rjmp IN_STRING              ;Repeat until NULL
character is reached.
END_IN:                         ret
```

PartF:

```
;
; lab4f.asm
;
; Created: 7/4/2016 3:54:55 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x3FFF
.equ bsel = 107
.equ bscale = -5

.dseg

Input_String: .byte 10
```

```
.cseg

.org 0x0000
        rjmp MAIN

.org USARTD0_RXC_VECT
        jmp ISR_RX

.org 0x0100


MAIN:
                                ldi YL, high(stack_init)        ;Initialize the stack pointer.
                                sts CPU_SPH, YL
                                ldi YL, low(stack_init)
                                sts CPU_SPL, YL
                                rcall USART_INIT                ;Initialize the USART.

                                ldi R16, 0xFF
                                sts PORTE_DIRSET, R16
                                rcall INIT_INT                  ;Call the routine to intialize
the interrupt.

TOGGLE:                                                         ;Infinite
Toggle while waiting for interrupt.
                                ldi R19, 0xAA
                                sts PORTE_OUTTGL, R19
                                rcall DELAY_x10ms
                                rjmp TOGGLE




USART_INIT:
                                ldi R16, 0x0A
                                sts PORTQ_DIRSET, R16           ;Set Port Q pins 1 and 3 to output.
                                sts     PORTQ_OUTCLR, R16       ;Output Defaul 0 to pins 1
and 3 of Port Q.
                                ldi R16, 0x08                   ;Set Pin 3 of Port D to
output (Tx).
                                sts PORTD_DIRSET, R16
                                sts PORTD_OUTSET, R16           ;Set the default value as 1.
                                ldi R16, 0x04
                                sts PORTD_DIRCLR, R16           ;Set RX pin as input.
                                ldi R16, 0x18
                                sts USARTD0_CTRLB, R16          ;Turn on the transmitter (Tx) and
receiver (Rx)
                                ldi R16, 0x10
                                sts USARTD0_CTRLA, R16          ;Turn on low level interrupt from Rx.
                                ldi R16, 0x03
                                sts USARTD0_CTRLC, R16          ;Async Mode, No Parity, 1 Stop bit,
8-bit Frame.
                                ldi R16, (bsel & 0xFF)          ;Lower 8-bits of the 12-bit bsel value
in register BAUDCTRLA
```

```
                                        sts USARTD0_BAUDCTRLA, R16
                                        ldi R16, ((bscale << 4) & 0xF0) | ((bsel >>  8) & 0x0F)
                                        sts USARTD0_BAUDCTRLB, R16      ;Most signification 4 bits of bsel
value in the least significant 4 bits of BAUDCTRLB.
                                        ret

INIT_INT:
                                        ldi R16, 0x01
                                        sts PMIC_CTRL, R16                              ;Turn on low-level
interrupts.
                                        sei
;Turn on global interrupt flag.
                                        ret

CHAR_OUT:
                                        push R17
TX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 5                                     ;If the DREIF flag
(Bit 5) is set then send to out or else wait until set.
                                        rjmp TX_POLL
                                        sts USARTD0_DATA, R16           ;Send the character in R16 out.
                                        pop R17
                                        ret

IN_CHAR:

RX_POLL:
                                        lds R17, USARTD0_STATUS         ;Load the USART status register.
                                        sbrs R17, 7                                     ;If the Rx Flag is
set then send to in or else wait until set.
                                        rjmp RX_POLL
                                        lds R16, USARTD0_DATA           ;Read the character into R16.
                                        ret




OUT_STRING:
                                        lpm R16, Z+                                     ;Load a value from
string table to R16.
                                        cpi R16, 0x00                           ;If the character is NULL
then end the output of the string.
                                        breq END_OUT
                                        rcall CHAR_OUT                          ;Output the next character
                                        rjmp OUT_STRING                         ;Repeat until NULL
character is reached.
END_OUT:                ret


IN_STRING:
                                        rcall IN_CHAR                           ;Read the input
                                        cpi R16, 0x00                           ;If the character is null then
end the reading of the string.
                                        breq END_IN
```

```
                                    st X+, R16                          ;Else store the
character in the location X is pointed to.
                                    rjmp IN_STRING                      ;Repeat until NULL
character is reached.
END_IN:                     ret


ISR_RX:
                                    push R19
                                    lds R19, CPU_SREG                   ;Push the status
register onto the stack.
                                    push R19
                                    rcall IN_CHAR                       ;Check what the
input was.
                                    nop
                                    rcall CHAR_OUT                      ;Echo the input back to the
output.
                                    ldi R16, 0x00
                                    sts PORTF_INTFLAGS, R16             ;Clear the interrupt flag.
                                    pop R19
                                    sts CPU_SREG, R19
                                    pop R19
                                    reti

DELAY_x10ms:
                                    ldi R16, 37                                 ;44 x
10ms = .44s
D_LOOP:                     rcall DELAY_10ms
                                    dec R16
                                    brne D_LOOP
                                    ret

DELAY_10ms:
                                    ldi R21, 79                                 ;.125ms
x 80 = 10ms
DELAY_LOOP:         ldi R20, 51                         ;25 cycles. .25uS x 50  =
.125ms
DELAY_p15ms:        dec R20
                                    nop
                                    nop
                                    nop
                                    brne DELAY_p15ms
                                    dec R21
                                    brne DELAY_LOOP
                                    ret
```

Appendix:

Switch Bounce.