Lab 5

Reyna, Ricardo

Section 75C9

07/09/2015

* Prelab Questions:

1. List the XMEGA's USART registers used in your programs and briefly describe their functions.

*UsartD0\_CTRLA: Is for interrupt level.*

*UsartD0\_CTRLB:Enables transmitter and receiver.*

*UsartD0\_CTRLC:Parity, stop bit, bit frame*

*USARTC0\_BAUDCTRLA & USARTC0\_BAUDCTRLB: Baud rate configurations.*

*USARTC0\_STATUS:Check the status of the RX and TX, to see if we can receive or transmit.*

*USARTC0\_DATA:To actually send or receive the data from/to the USARTC system*

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

*Synchronous have a clock that was to wait in order to do something. While asynchronous doesn't have to wait and can be done at any time. In the lab we used asynchronous communication.*

3. What's the difference between serial and parallel communication?

*Serial communication transfers data one bit at a time, uses less wires, and can cover longer distances. Parallel communication sends data multiple bits at a time over multiple wires, and can do it much quicker.*

4. List the number of bounces from part A of this lab. How long (in ms) is your delay routine for debouncing?

*There are around 10 bounces on falling, and 2 bounces on rising. I used a 90ms delay for the debouncing.*

5. What is the maximum possible baud you can use **for asynchronous communication** if your board runs at 2Mhz? Support your answer with the values you would place in any special registers that are needed.

*he maximum baud rate is 124031 Hz, the bscale would be -7 and 1 for bsel.*

ldi R16, (BSel & 0xFF)

sts USARTD0\_BAUDCTRLA, R16

ldi R16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F) sts USARTD0\_BAUDCTRLB, R16

* Problems Encountered:

Getting the menu to pop before taking in any characters into the console.

* Future Work/Applications:

We now know how to interrupt a service which is useful for Operating Systems or other devices. We also learned serial communication which is used to receive and send data.

* Schematics:

None for this lab

* Decoding Logic:

None for this lab

* Pseudo code:

Part A:

Start a register with 0

Infinite loop

inside the interrupt:

increase register

output to led

PartC:

Fill in table with favorite stuff

Make an infinite loop where depending and the character it makes an rcall

when esc char is pressed end program

PartD:

Infinite loop turning LED on and off every .37s

Inside interrupt:  
A get char

and spit it out in the console

* Program Code:

Part A:

(Note: The two include files are at the end)

/\*

\* Lab5\_A\_RSR.asm

\*

\* Created: 7/2/2015 1:26:06 PM

\* Author: stefano92

Lab 5 part A

Name: Ricardo Stefano Reyna

Section#: 75C9

TA: Khaled Hassan

Description: This program will count the number of interrupts and display them on the LED

\*/

.include "atxmega128a1udef.inc"

.include "EBI\_STUFF.asm"

.include "Delay.asm"

.org 0x0000

rjmp MAIN

.org PORTE\_INT0\_VECT ;place code at the interrupt vector for the PORTE\_INT0 interrupt

rjmp EXT\_INT\_ISR ;jump to our interrupt routine

.org 0x0200

MAIN:

STACK\_STUFF ;Initialize stack

EBI\_INIT ;Initialize EBI

CS0\_INIT ;Initialize CS0

/////////////////////////START PROGRAM///////////////////

ldi r17, 0x00

rcall INIT\_INTERRUPT ;call our subroutine to initialize our interrupt

nop

LOOP:

rjmp LOOP ;loop forver while our interrupt fires

INIT\_INTERRUPT:

ldi r16, 0x01 ;select PORTE\_PIN0 as the interrupt source

sts PORTE\_INT0MASK, r16

sts PORTE\_DIRCLR, r16 ;PIN0 as input

ldi r16, 0x01 ;select the external interrupt as a low level

sts PORTE\_INTCTRL, r16 ; priority interrupt

; Probably inappropriately cleared the INT1 interrupt level pins

ldi R16, 0x01 ;select low level pin for external interrupt

sts PORTE\_PIN0CTRL, r16 ; (rising edge)

; Probably inappropriately cleared pins 7, 5, 4, 3

ldi r16, 0x01

sts PMIC\_CTRL, r16 ;turn on low level interrupts

; Also effected pins 7-1

sei ;turn on the global interrupt flag LAST!

ret

EXT\_INT\_ISR:

call WASTE

push r16

in r16, CPU\_SREG

push r16

nop ;dummy instruction to put a breakpoint on

inc r17

st X, r17 ;Number of interrupts

ldi r16, 0x01

sts PORTE\_INTFLAGS, r16 ; Clear the PORTE\_INTFLAGS

pop r16

out CPU\_SREG, r16

pop r16

reti ;return from the interrupt routine

PartC:

/\*

\* Lab5\_C\_RSR.asm

\*

\* Created: 7/6/2015 11:03:15 PM

\* Author: stefano92

Lab 5 part C

Name: Ricardo Stefano Reyna

Section#: 75C9

TA: Khaled Hassan

Description: This program will print strings of my favorite stuff

\*/

.include "atxmega128a1udef.inc"

.include "EBI\_STUFF.asm"

.equ BSel = 983

.equ BScale = -7 ;14400Hz

.equ CR = 0x0D

.equ LF = 0x0A

.equ ESC = 0x1B

.org 0x1000

MENU: .db "Stefano's favorite:", CR, LF, "1. Movie", CR, LF, "2. Book", CR, LF, "3. Food", CR, LF, "4. Ice Cream/Yogurt flavor", CR, LF, "5. Pizza Topping", CR, LF, "6. Redisplay Menu", CR, LF, "ESC: exit", CR, LF, 0x00

OP1: .db "Stefano's favorite movie is Fight Club", CR, LF, 0x00

OP2: .db "Stefano's favorite book is Chronicles of a Death Fortold", CR, LF, 0x00

OP3: .db "Stefano's favorite food is rice", CR, LF, 0x00

OP4: .db "Stefano's favorite ice cream/yogurt flavor is chocolate", CR, LF, 0x00

OP5: .db "Stefano's favorite pizza topping is pineapple", CR, LF, 0x00

OP6: .db "Done!", 0x00

.org 0x0000

rjmp MAIN

.org 0x0200

MAIN:

STACK\_STUFF

rcall INIT\_USART

rcall INIT\_GPIO

LOOP:

ldi ZL, low(MENU << 1)

ldi ZH, high(MENU << 1)

rcall OUT\_STRING

GETCHAR:

rcall IN\_CHAR

cpi R16, ESC ;If ESC

breq EXIT

cpi R16, 0x31 ;If 1

breq MOVIE

cpi R16, 0x32 ;If 2

breq BOOK

cpi R16, 0x33 ;If 3

breq FOOD

cpi R16, 0x34 ;If 4

breq ICEYO

cpi R16, 0x35 ;If 5

breq PIZZA

cpi R16, 0x36 ;If 6

breq LOOP

rjmp GETCHAR

;OPTIONS

EXIT:

ldi ZL, low(OP6 << 1)

ldi ZH, high(OP6 << 1)

call OUT\_STRING

DONE:

rjmp DONE

MOVIE:

ldi ZL, low(OP1 << 1)

ldi ZH, high(OP1 << 1)

rcall OUT\_STRING

rjmp LOOP

BOOK:

ldi ZL, low(OP2 << 1)

ldi ZH, high(OP2 << 1)

rcall OUT\_STRING

rjmp LOOP

FOOD:

ldi ZL, low(OP3 << 1)

ldi ZH, high(OP3 << 1)

rcall OUT\_STRING

rjmp LOOP

ICEYO:

ldi ZL, low(OP4 << 1)

ldi ZH, high(OP4 << 1)

rcall OUT\_STRING

rjmp LOOP

PIZZA:

ldi ZL, low(OP5 << 1)

ldi ZH, high(OP5 << 1)

rcall OUT\_STRING

rjmp LOOP

;OUTSTR

OUT\_STRING:

push r16

WRITE:

lpm r16, Z+ ;reads each char

cpi r16, 0x00

breq STOPW

rcall OUT\_CHAR

rjmp WRITE ;Write to console

STOPW:

pop r16

ret

;INCHAR

IN\_CHAR:

push r17

RX\_POLL:

lds r16, USARTD0\_STATUS ;load the status register

sbrs r16, 7 ;proceed to reading in a char if

; the receive flag is set

rjmp RX\_POLL ;else continue polling

lds r16, USARTD0\_DATA ;read the character into R16

pop r17

ret

;OUTCHAR

OUT\_CHAR:

push R17

TX\_POLL:

lds R17, USARTD0\_STATUS ;load status register

sbrs R17, 5 ;proceed to writing out the char if

; the DREIF flag is set

rjmp TX\_POLL ;else go back to polling

sts USARTD0\_DATA, R16 ;send the character out over the USART

pop R17

ret

;INITUSART

INIT\_USART:

ldi R16, 0x18

sts USARTD0\_CTRLB, R16 ;turn on TXEN, RXEN lines

ldi R16, 0x03

sts USARTD0\_CTRLC, R16 ;Set Parity to none, 8 bit frame, 1 stop bit

ldi R16, (BSel & 0xFF) ;select only the lower 8 bits of BSel

sts USARTD0\_BAUDCTRLA, R16 ;set baudctrla to lower 8 bites of BSel

ldi R16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)

sts USARTD0\_BAUDCTRLB, R16 ;set baudctrlb to BScale | BSel. Lower

; 4 bits are upper 4 bits of BSel

; and upper 4 bits are the BScale.

ret

;INITGPIO

INIT\_GPIO:

ldi R16, 0x08

sts PortD\_DIRSET, R16 ;Must set PortD\_PIN3 as output for TX pin

; of USARTD0

sts PortD\_OUTSET, R16 ;set the TX line to default to '1' as

; described in the documentation

ldi R16, 0x04

sts PortD\_DIRCLR, R16 ;Set RX pin for input

ldi R16, 0xA ; PortQ bits 1 and 3 enable and select

sts PORTQ\_DIRSET, R16 ; the PortD bits 2 and 3 serial pins

sts PORTQ\_OUTCLR, R16 ; to be connected to the USB lines

ret

PartD:

/\*

\* Lab5\_D\_RSR.asm

\*

\* Created: 7/8/2015 1:21:39 PM

\* Author: stefano92

Lab 5 part D

Name: Ricardo Stefano Reyna

Section#: 75C9

TA: Khaled Hassan

Description: This program combines part A and C

\*/

.include "atxmega128a1udef.inc"

.include "EBI\_STUFF.asm"

.include "Delay.asm"

.equ BSel = 983

.equ BScale = -7 ;14400Hz

.org 0x0000

rjmp MAIN

.org USARTD0\_RXC\_vect ;place code at the interrupt vector for the PORTE\_INT0 interrupt

rjmp EXT\_INT\_ISR ;jump to our interrupt routine

.org 0x0200

MAIN:

STACK\_STUFF ;Initialize stack

EBI\_INIT ;Initialize EBI

CS0\_INIT ;Initialize CS0

rcall INIT\_USART

rcall INIT\_GPIO

ldi r16, 0x04

ldi r18, 0x00

LOOP:

ldi r19, 74

st X, r16 ;ON

call WASTE

st X, r18 ;OFF

call WASTE

rjmp LOOP

;INCHAR

IN\_CHAR:

push r17

RX\_POLL:

lds r16, USARTD0\_STATUS ;load the status register

sbrs r16, 7 ;proceed to reading in a char if

; the receive flag is set

rjmp RX\_POLL ;else continue polling

lds r16, USARTD0\_DATA ;read the character into R16

pop r17

ret

;OUTCHAR

OUT\_CHAR:

push R17

TX\_POLL:

lds R17, USARTD0\_STATUS ;load status register

sbrs R17, 5 ;proceed to writing out the char if

; the DREIF flag is set

rjmp TX\_POLL ;else go back to polling

sts USARTD0\_DATA, R16 ;send the character out over the USART

pop R17

ret

;INITUSART

INIT\_USART:

ldi r16, 0x10

sts USARTD0\_CTRLA, r16 ;turn on low level

ldi r16, 0x18

sts USARTD0\_CTRLB, r16 ;turn on TXEN, RXEN lines

ldi r16, 0x03

sts USARTD0\_CTRLC, r16 ;Set Parity to none, 8 bit frame, 1 stop bit

ldi r16, (BSel & 0xFF) ;select only the lower 8 bits of BSel

sts USARTD0\_BAUDCTRLA, r16 ;set baudctrla to lower 8 bites of BSel

ldi r16, ((BScale << 4) & 0xF0) | ((BSel >> 8) & 0x0F)

sts USARTD0\_BAUDCTRLB, r16 ;set baudctrlb to BScale | BSel. Lower

; 4 bits are upper 4 bits of BSel

; and upper 4 bits are the BScale.

ldi r16, 0x01

sts PMIC\_CTRL, r16 ;turn low level interrupts ON

sei ;set the global interrupt flag to enable interrupt

ret

;INITGPIO

INIT\_GPIO:

ldi R16, 0x08

sts PortD\_DIRSET, R16 ;Must set PortD\_PIN3 as output for TX pin

; of USARTD0

sts PortD\_OUTSET, R16 ;set the TX line to default to '1' as

; described in the documentation

ldi R16, 0x04

sts PortD\_DIRCLR, R16 ;Set RX pin for input

ldi R16, 0xA ; PortQ bits 1 and 3 enable and select

sts PORTQ\_DIRSET, R16 ; the PortD bits 2 and 3 serial pins

sts PORTQ\_OUTCLR, R16 ; to be connected to the USB lines

ret

EXT\_INT\_ISR:

push r16

in r16, CPU\_SREG

push r16

nop ;dummy instruction to put a breakpoint on

rcall IN\_CHAR ;Read char

rcall OUT\_CHAR ;Output char

pop r16

out CPU\_SREG, r16

pop r16

reti ;return from the interrupt routine

Delay:

;This one takes r19 as an input

.LIST

.org 0xF00

WASTE:

push r16

push r17

ldi r16, 0x00

ldi r17, 0x00

EXTRA:

inc r16

cp r16, r19 ;change this number

brne DELAY\_10ms

pop r17

pop r16

ret

DELAY\_10ms:

nop ; do nothing

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

inc r17 ;increase r18

cpi r17, 0xFF ;check if r18 is 28

brne DELAY\_10ms ;if not loop

rjmp EXTRA

EBI\_STUFF:

;Simple program to initialize stuff

.set IOPORT = 0x4000

.set SRAMPORT = 0x1B0000

.macro STACK\_STUFF

ldi r16, 0xFF

out CPU\_SPL, r16 ;initialize low byte of stack pointer

ldi r16, 0x3F

out CPU\_SPH, r16 ;initialize high byte of stack pointer

.endmacro

.macro EBI\_INIT

ldi r16, 0x17

sts PORTH\_DIR, r16 //set port pins as outputs for RE and ALE and WE

ldi r16, 0x13

sts PORTH\_OUT, r16 //WE and RE is active low so it must be set

ldi r16, 0xFF

sts PORTJ\_DIR, r16 //set datalines as outputs

sts PORTK\_DIR, r16 //set address lines as outputs

ldi r16, 0x01

sts EBI\_CTRL, r16 //turn on 3 port SRAM ALE1 EBI

.endmacro

.macro CS0\_INIT

ldi ZH, HIGH(EBI\_CS0\_BASEADDR) //all the set up for CS0, since EBI won't work without it

ldi ZL, LOW(EBI\_CS0\_BASEADDR)

ldi r16, ((IOPORT>>8) & 0xF0)

st Z+, r16

ldi r16, ((IOPORT>>16) & 0xFF)

st Z, r16

ldi r16, 0x11

sts EBI\_CS0\_CTRLA, r16

ldi XH, HIGH(IOPORT)

ldi XL, LOW(IOPORT)

.endmacro

.macro CS1\_INIT

ldi ZH, HIGH(EBI\_CS1\_BASEADDR) //set up CS1 for the SRAM

ldi ZL, LOW(EBI\_CS1\_BASEADDR)

ldi r16, ((SRAMPORT>>8) & 0xF0)

st Z+, r16

ldi r16, ((SRAMPORT>>16) & 0xFF)

st Z, r16

ldi r16, 0b00011101

sts EBI\_CS1\_CTRLA, r16

.endmacro

* Appendix:

Falling edge  


Rising Edge

