Brandon Pollack

Lab 5 report

All ASM files including the includes

EBI_INIT.asm

```
/* A collection of inits for the EBI and stack init at the bottom */
.set IOPORT = 0x5000
.set SRAMPORT = 0x370000


.macro TRIPORT_ALE_ONE_INIT


        ldi R16, 0b00110111
        sts PORTH_DIR, R16 //set port pins as outputs for RE and ALE and WE CS1 and CS0

        ldi R16, 0b00110011
        sts PORTH_OUTSET, R16 //WE and RE and CS pins are active low so it must be set

        ldi R16, 0xFF
        sts PORTJ_DIR, R16 //set datalines as outputs (manual says so)
        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
.endmacro

.macro CS1_INIT
        .equ SRAMPORT = 370000
```

```
	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


.macro STACK_INIT
	ldi R16, 0xFF
	out CPU_SPL, R16
	ldi R16, 0x3F
	out CPU_SPH, R16 //init stack pointer
.endmacro
```

UART_INITS

```
/* Brandon Pollack
*  HW4
*  SCI Subroutines
*/


.macro SCI_C_INIT

	.equ BSEL = 51
	.equ BSCL = -2

	ldi R16, 0x18
	sts USARTC0_CTRLB, R16		;this buts a one in RXEN and TXEN, enabling
transmission and receive

	ldi R16, 0x03
	sts USARTC0_CTRLC, R16		;No parity, 8 bit data, a single stop bit

	ldi R16, BSEL
	sts USARTC0_BAUDCTRLA, R16	;setting baud to 9600 HZ involves some calculation from
the manual
```

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

        sts USARTC0_BAUDCTRLB, R16     ;set the scale to -2 as per the formula to get 9600 HZ
from the Fper and BSCL, upper 4 bits of BSEL stay the same

        ; now begins the set up of the PORTC to output and input serial

        ldi R16, 0x08
        sts PORTC_DIR, R16
        sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs
.endmacro
/*SPI_C_INIT:
        .macro
        .equ BSEL = 51
        .equ BSCL = -2


        ldi R16, 0x18
        sts USARTC0_CTRLB, R16        ;this buts a one in RXEN and TXEN, enabling
transmission and receive

        ldi R16, 0b01000011
        sts USARTC0_CTRLC, R16        ;No parity, 8 bit data, a single stop bit, synchronous
transmission

        ldi R16, BSEL
        sts USARTC0_BAUDCTRLA, R16    ;setting baud to 9600 HZ involves some calculation from
the manual

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

        sts USARTC0_BAUDCTRLB, R16    ;set the scale to -2 as per the formula to get 9600 HZ
from the Fper and BSCL, upper 4 bits of BSEL stay the same

        ; now begins the set up of the PORTC to output and input serial

        ldi R16, 0x08
        sts PORTC_DIR, R16
        sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs
        .endmacro
        */

OUT_CHAR:
        .org 0x200
        push R17 ;save this value

        isdatasent:
```

```
		lds R17, USARTC0_STATUS
		sbrs R17, 6 ;poll TXIF in status register, if it is clear we are not done
		rjmp isdatasent

	sts USARTC0_DATA, R16

	pop R17

	ret

OUT_STRING:
	push R16 ;I chose to use z so this sub works for program or data memory (remember to
shift left if program memory)

	beginwritingstring:
		ld R16, Z+ ;at the end of this sub, z will point to one address past the end
of the string
		breq donewritingstring
		call OUT_CHAR
		rjmp beginwritingstring

	donewritingstring:
		pop R16
		ret

IN_CHAR:
	push R17

	isdatarecieved:
		lds R17, USARTC0_STATUS
		sbrs R17, 7
		rjmp isdatarecieved

	lds R16, USARTC0_DATA

	pop R17
	ret

IN_STRING:  ;be sure to have X point where you want this data to go
	push R16

	beginreadingstring:
		call IN_CHAR ;puts the character in R16
		cpi R16, 0
		breq donereadingstring
		st X+, R16
		rjmp beginreadingstring

	donereadingstring:
        pop R16
        ret
```

Lab5_part1_BRP

```
/*
 * Brandon Pollack
 * Ivan
 * 1524
 * A program that uses an input on a port as an interrupt
 */
 .include "Atxmega128A1udef.inc"
 .include "EBI_INITS.asm"
.org 0
        rjmp main

.org PORTC_INT0_VECT
        rjmp EXT_INT_countup

main:
.org 0x100
        ldi R16, 0x01
        sts PORTC_INTCTRL, R16 ;set this port as a low level interrupt

        ldi R16, 0x04
        sts PORTC_INT0MASK, R16 ;set pin 2 as the interrupt, since it is the only one with
full asynch support
        sts PORTC_DIRCLR, R16 ;make certain that pin is an input

        ldi R16, 0x02
        sts PORTC_PIN2CTRL, R16 ;set pin 2 to trigger an interrupt on only a falling edge

        ldi R16, 0x01
        sts PMIC_CTRL, R16 ;turn on low level interrupts

        sei ;turn on interrupts

        TRIPORT_ALE_ONE_INIT ;turn on EBI so I can write to my LEDs
        CS0_INIT ;not really needed but I think my CPLD uses it now so lets turn it on
        ldi R16, 0x0 ;set the init value of our count to 0
        sts IOPORT, R16

loopforever:
        rjmp loopforever

EXT_INT_countup:
```

```
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
nop
nop

ldi R17, 0x00
sts PORTC_INTFLAGS, R17
inc R16
sts IOPORT, R16 ;simply increments R16 and writes it out to the IO ports

reti
```

Lab5_part2_1

```
/*
 *Brandon Pollack
 *Ivan
 *1352
 *UART Port C program to interface with a terminal
 */

.include "Atxmega128A1udef.inc"
.include "EBI_INITS.asm"

.equ CR = 0x0D
.equ LF = 0x0A
.equ stringlocation = 0x1000

.org stringlocation
	.DB "My name is Brandon Pollack, my favourite movie is Pulp Fiction, my favourite
class is EEL4744, my favourite TV show is Star Trek." CR, LF, "Instructor: Dr. Eric M.
Schwartz, TA: IVAN", CR, LF, 0x00


.org 0x0
	rjmp main

main:
.org 0x100

	STACK_INIT

	.equ BSEL = 144
	.equ BSCL = -6

	ldi R16, 0x18
	sts USARTC0_CTRLB, R16		;this buts a one in RXEN and TXEN, enabling
transmission and receive

	ldi R16, 0x03
	sts USARTC0_CTRLC, R16		;No parity, 8 bit data, a single stop bit

	ldi R16, BSEL
	sts USARTC0_BAUDCTRLA, R16	;setting baud to 9600 HZ involves some calculation from
the manual

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

	sts USARTC0_BAUDCTRLB, R16	;set the scale to -2 as per the formula to get 9600 HZ
from the Fper and BSCL, upper 4 bits of BSEL stay the same
```

```
	; now begins the set up of the PORTC to output and input serial

	ldi R16, 0x08
	sts PORTC_DIR, R16
	sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs


	ldi ZL, low(stringlocation << 1)
	ldi ZH, high(stringlocation << 1) ;string location shifted by one

	call OUT_STRING

done: rjmp done



OUT_CHAR:
	.org 0x200
	push R17 ;save this value

	isdatasent:
		lds R17, USARTC0_STATUS
		sbrs R17, 5 ;poll TXIF in status register, if it is clear we are not done
		rjmp isdatasent

	sts USARTC0_DATA, R16

	pop R17

	ret

OUT_STRING:
	push R16 ;I chose to use z so this sub works for program or data memory (remember to
shift left if program memory)

	beginwritingstring:
		lpm R16, Z+ ;at the end of this sub, z will point to one address past the end
of the string
		cpi R16, 0x0
		breq donewritingstring
		call OUT_CHAR
		rjmp beginwritingstring

	donewritingstring:
		pop R16
		ret
```

Lab5_part2_2

```
/*
 * Brandon Pollack
 * Ivan
 * 1524
 * A program that uses interrupts to manage UART
 */
 .include "Atxmega128A1udef.inc"
 .include "EBI_INITS.asm"

.org 0x0
rjmp main

.org USARTC0_RXC_vect
	rjmp RX_ISR

.org 0x100
main:
	.equ BSEL = 144
	.equ BSCL = -6

	ldi R16, 0x18
	sts USARTC0_CTRLB, R16		;this buts a one in RXEN and TXEN, enabling
transmission and receive

	ldi R16, 0x03
	sts USARTC0_CTRLC, R16		;No parity, 8 bit data, a single stop bit

	ldi R16, BSEL
	sts USARTC0_BAUDCTRLA, R16	;setting baud to involves some calculation from the
manual

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

	sts USARTC0_BAUDCTRLB, R16	;set the scale and last 4 bits of BSEL

	; now begins the set up of the PORTC to output and input serial

	ldi R16, 0x08
	sts PORTC_DIR, R16
	sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs

	//done with UART setup (except Interrupt drives, whcih I will do after I initilize
PMIC)

	ldi R16, 0x01
	sts PMIC_CTRL, R16 //globally enable low level interrupts
```

```
        ldi R16, (0x01 << 4)
        sts USARTC0_CTRLA, R16 //set RXINTLVL to a low level interrupt

        sei //globally enable all interrupts

        TRIPORT_ALE_ONE_INIT
        CS0_INIT
        STACK_INIT
        ldi XL, low(IOPORT)
        ldi XH, high(IOPORT)
        ldi R16, 0xFF
        st X, R16

loopforever:
        ldi R16, 0xFF
        st X, R16
        call DELAY500
        ldi R16, 0x00
        st X, R16
        call DELAY500
        rjmp loopforever

RX_ISR:
        push R17
        push R16
        lds R16, USARTC0_DATA
        isdatasent:
                lds R17, USARTC0_STATUS
                sbrs R17, 5 ;poll DATA in status register, if it is clear we are not done
                rjmp isdatasent

        sts USARTC0_DATA, R16
        pop R16
        pop R17
        reti

DELAY500:
push R16
push R17

ldi R16, 0
ldi R17, 0

AGAIN:
NOP
NOP
NOP
NOP
```

```
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
INC R16
CPI R16, 0
BREQ CARRY

BACK:
CPI R17, 0xFF
BRNE AGAIN
BREQ RETURN

CARRY:
INC R17
rjmp BACK

RETURN:
        pop R17
        pop R16
        RET
```

Lab5_part2_3

```
/*
 * Lab5_part2_3.asm
 *
 *  Created: 3/3/2013 3:41:22 PM
 *   Author: Brandon
 *Brandon Pollack
 *Ivan
 *1352
 *UART Port C program to interface with a terminal, now taking input
 */

.include "Atxmega128A1udef.inc"
.include "EBI_INITS.asm"

.equ CR = 0x0D
.equ LF = 0x0A
.equ menulocation = 0x1000
.equ TAB = 0x09
.equ CC = 0x12

.org menulocation
	.db "Brandon's Favourite:",CR,LF,"0:",  TAB,"Sport",CR,LF,"1:",TAB,"TV
Show",CR,LF,"2:",TAB,"Book",CR,LF,"3:",TAB,"Food",CR,LF,"4:",TAB,"Movie",CR,LF,"5:",TAB,"Disp
lay menu",CR,LF,"ESC: exit",CR,LF,0x00

fsport:
	.db "Baseball",CR,LF,0
fTV:
	.db "Star Trek",CR,LF,0
fBook:
	.db "Ender's Game",CR,LF,0
fFood:
	.db "Tonkotsu Ramen",CR,LF,0
fMovie:
	.db "Pulp Fiction",CR,LF,0
exitprint:
	.db "Done!",0



.org 0x0
	rjmp main

main:
.org 0x100

	STACK_INIT
```

```
        .equ BSEL = 144
        .equ BSCL = -6

        ldi R16, 0x18
        sts USARTC0_CTRLB, R16        ;this buts a one in RXEN and TXEN, enabling
transmission and receive

        ldi R16, 0x03
        sts USARTC0_CTRLC, R16        ;No parity, 8 bit data, a single stop bit

        ldi R16, BSEL
        sts USARTC0_BAUDCTRLA, R16    ;setting baud to 9600 HZ involves some calculation from
the manual

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

        sts USARTC0_BAUDCTRLB, R16    ;set the scale to -2 as per the formula to get 9600 HZ
from the Fper and BSCL, upper 4 bits of BSEL stay the same

        ; now begins the set up of the PORTC to output and input serial

        ldi R16, 0x08
        sts PORTC_DIR, R16
        sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs




loop:
        ldi ZL, low(menulocation << 1)
        ldi ZH, high(menulocation << 1) ;string location shifted by one
        call OUT_STRING
recievenewchar:
        call IN_CHAR

        cpi R16, 0x1B
        breq exitroutine
        cpi R16, '0'
        breq sport
        cpi R16, '1'
        breq TVshow
        cpi R16, '2'
        breq book
        cpi R16, '3'
        breq food
        cpi R16, '4'
        breq movie
        cpi R16, '5'
        breq loop
        rjmp recievenewchar

exitroutine:
        ldi ZL, low(exitprint << 1)
        ldi ZH, high(exitprint << 1)
        call OUT_STRING
done:
        rjmp done

sport:
        ldi R16, CC
        call OUT_CHAR
        ldi ZL, low(fsport << 1)
        ldi ZH, high(fsport << 1)
        call OUT_STRING
        rjmp loop
TVshow:
        ldi R16, CC
        call OUT_CHAR
        ldi ZL, low(fTV << 1)
        ldi ZH, high(fTV << 1)
        call OUT_STRING
        rjmp loop
book:
        ldi R16, CC
        call OUT_CHAR
        ldi ZL, low(fbook << 1)
        ldi ZH, high(fbook << 1)
        call OUT_STRING
        rjmp loop
food:
        ldi R16, CC
        call OUT_CHAR
        ldi ZL, low(fFood << 1)
        ldi ZH, high(fFood << 1)
        call OUT_STRING
        rjmp loop
movie:
        ldi R16, CC
        call OUT_CHAR
        ldi ZL, low(fmovie << 1)
        ldi ZH, high(fmovie << 1)
        call OUT_STRING
        rjmp loop

IN_CHAR:
        push R17
        isdatarecieved:
                lds R17, USARTC0_STATUS
                sbrs R17, 7
                rjmp isdatarecieved

        lds R16, USARTC0_DATA

        pop R17
        ret

OUT_CHAR:
        push R17 ;save this value

        isdatasent:
                lds R17, USARTC0_STATUS
                sbrs R17, 5 ;poll TXIF in status register, if it is clear we are not done
                rjmp isdatasent

        sts USARTC0_DATA, R16

        pop R17

        ret

OUT_STRING:
        push R16 ;I chose to use z so this sub works for program or data memory (remember to
shift left if program memory)

        beginwritingstring:
                lpm R16, Z+ ;at the end of this sub, z will point to one address past the end
of the string
                cpi R16, 0x0
                breq donewritingstring
                call OUT_CHAR
                rjmp beginwritingstring

        donewritingstring:
                pop R16
                ret
```

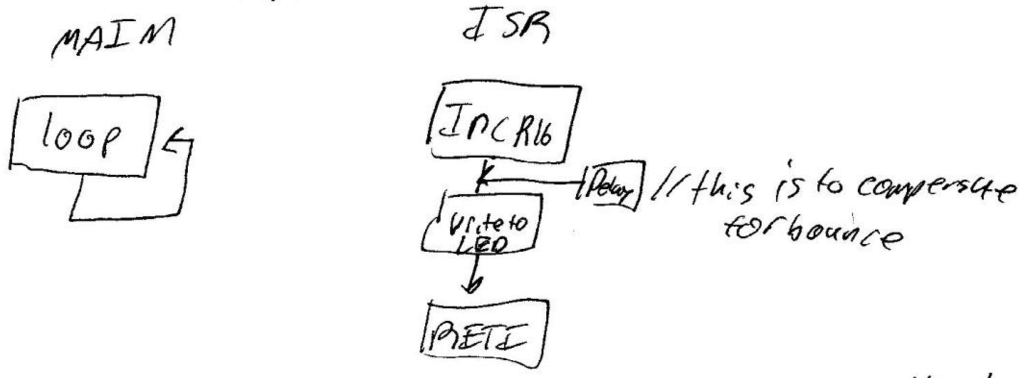
Lab5_part3

```
/*
 * Brandon Pollack
 * Ivan
 * 1524
 * A program that uses interrupts to manage UART
 */
 .include "Atxmega128A1udef.inc"
 .include "EBI_INITS.asm"

.org 0x0
rjmp main

.org USARTC0_RXC_vect
	rjmp RX_ISR

.org 0x100
main:
	.equ BSEL = 144
	.equ BSCL = -6

	ldi R16, 0x18
	sts USARTC0_CTRLB, R16		;this buts a one in RXEN and TXEN, enabling
transmission and receive

	ldi R16, 0x03
	sts USARTC0_CTRLC, R16		;No parity, 8 bit data, a single stop bit

	ldi R16, BSEL
	sts USARTC0_BAUDCTRLA, R16	;setting baud to involves some calculation from the
manual

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

	sts USARTC0_BAUDCTRLB, R16	;set the scale and last 4 bits of BSEL

	; now begins the set up of the PORTC to output and input serial

	ldi R16, 0x08
	sts PORTC_DIR, R16
	sts PORTC_OUT, R16 ; set the direction of the TX line as out and default as 1 as per
docs

	//done with UART setup (except Interrupt drives, whcih I will do after I initilize
PMIC)

	ldi R16, 0x01
	sts PMIC_CTRL, R16 //globally enable low level interrupts
```

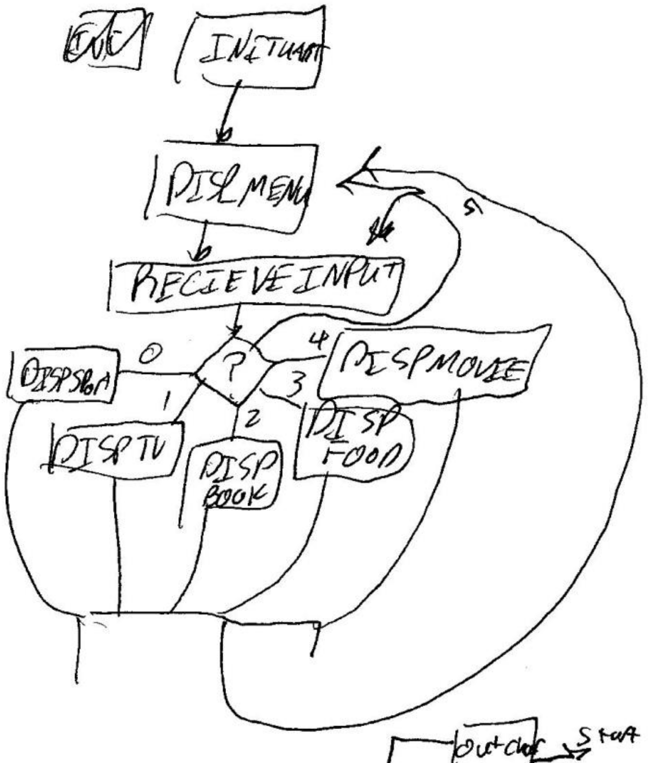
```
        ldi R16, (0x01 << 4)
        sts USARTC0_CTRLA, R16 //set RXINTLVL to a low level interrupt

        sei //globally enable all interrupts

        TRIPORT_ALE_ONE_INIT
        CS0_INIT
        STACK_INIT
        ldi XL, low(IOPORT)
        ldi XH, high(IOPORT)
        ldi R16, 0xFF
        st X, R16

loopforever:
        ldi R16, 0xFF
        st X, R16
        call DELAY500
        ldi R16, 0x00
        st X, R16
        call DELAY500
        rjmp loopforever

RX_ISR:
        push R17
        push R16
        lds R16, USARTC0_DATA
        isdatasent:
                lds R17, USARTC0_STATUS
                sbrs R17, 5 ;poll DATA in status register, if it is clear we are not done
                rjmp isdatasent

        sts USARTC0_DATA, R16
        pop R16
        pop R17
        reti

DELAY500:
push R16
push R17

ldi R16, 0
ldi R17, 0

AGAIN:
NOP
NOP
NOP
NOP
```

```
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
INC R16
CPI R16, 0
BREQ CARRY

BACK:
CPI R17, 0xFF
BRNE AGAIN
BREQ RETURN

CARRY:
INC R17
rjmp BACK

RETURN:
        pop R17
        pop R16
        RET
```

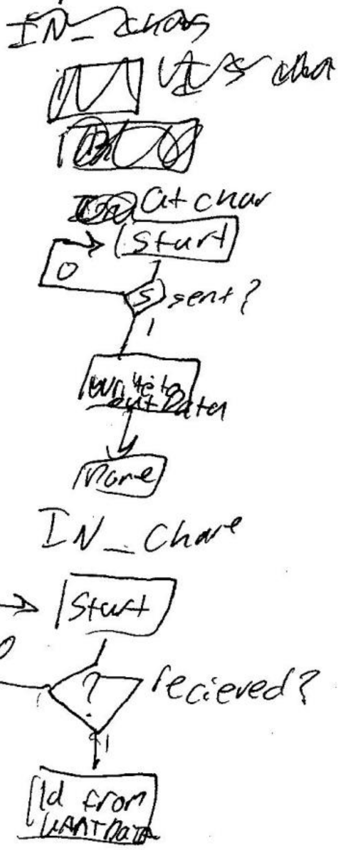
Part 1

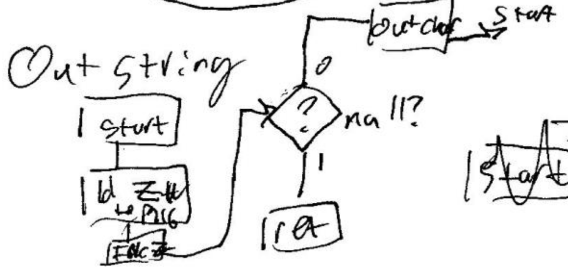


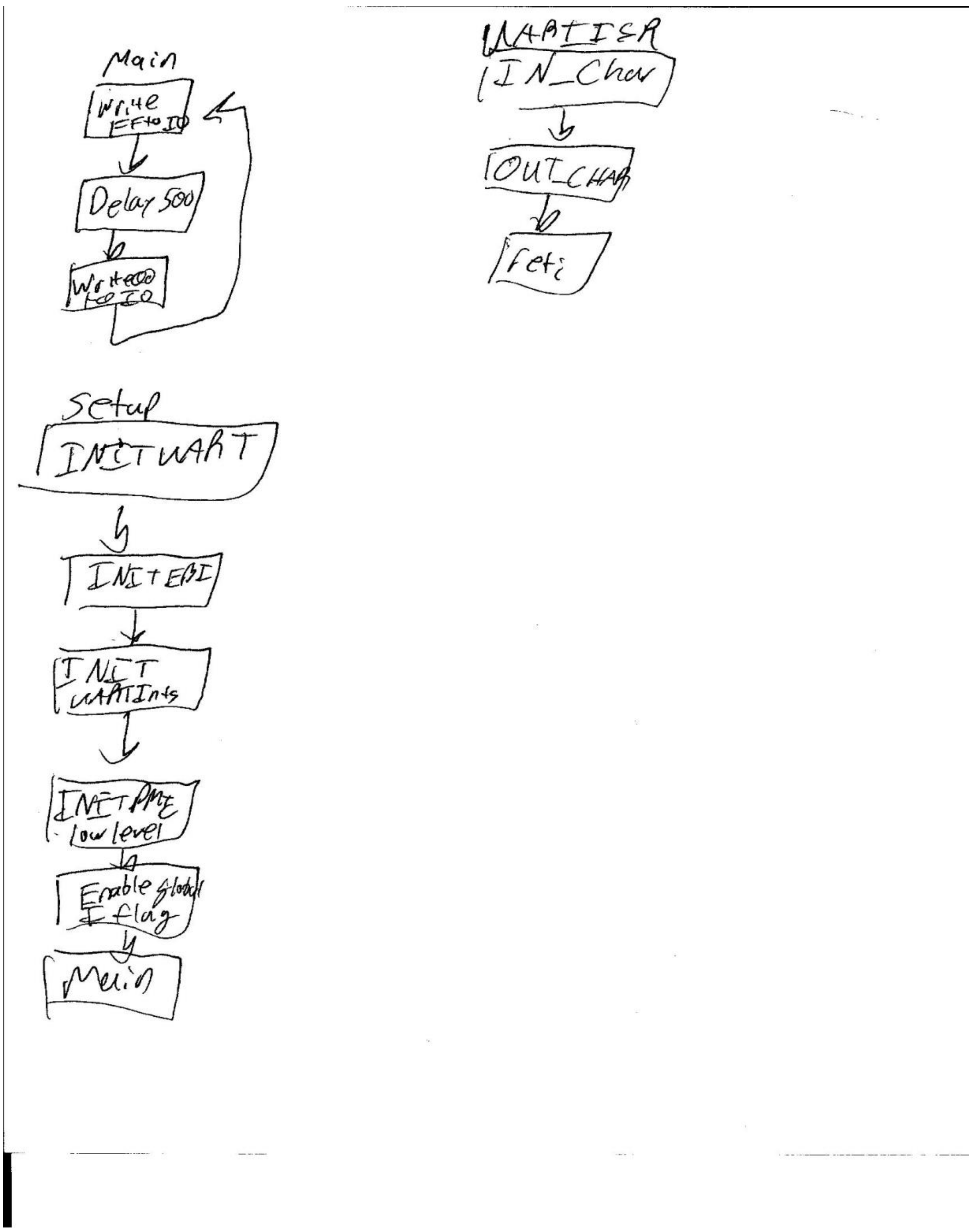
Part 2_2



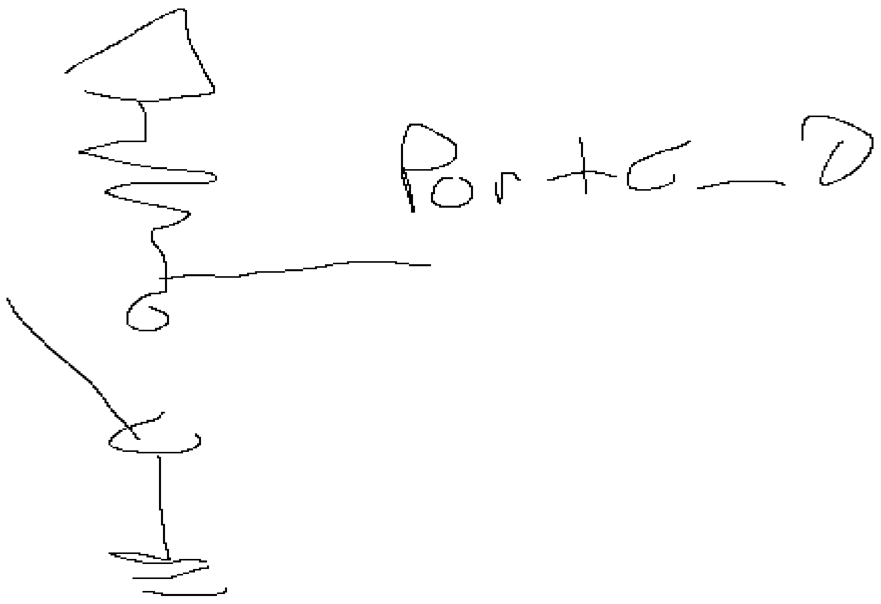
PART 2_~~1~~






Main
Write FF to IO
Delay 500
Write 00 to IO
Setup
INIT UART
INIT EBI
INIT UARTInts
INIT PME low level
Enable global I flag
Main
UARTISR
IN_Char
OUT_CHAR
reti

Part 3



Interrupt port