Aviles, Jean-Ralph
EEL3744
Section 1539
September 22nd, 2015
Lab 2

Problems Encountered

I messed up big time with my wire wrapping, broke a header and wire wrapped the switches and leds to the wrong ports. A lot of wasted time.

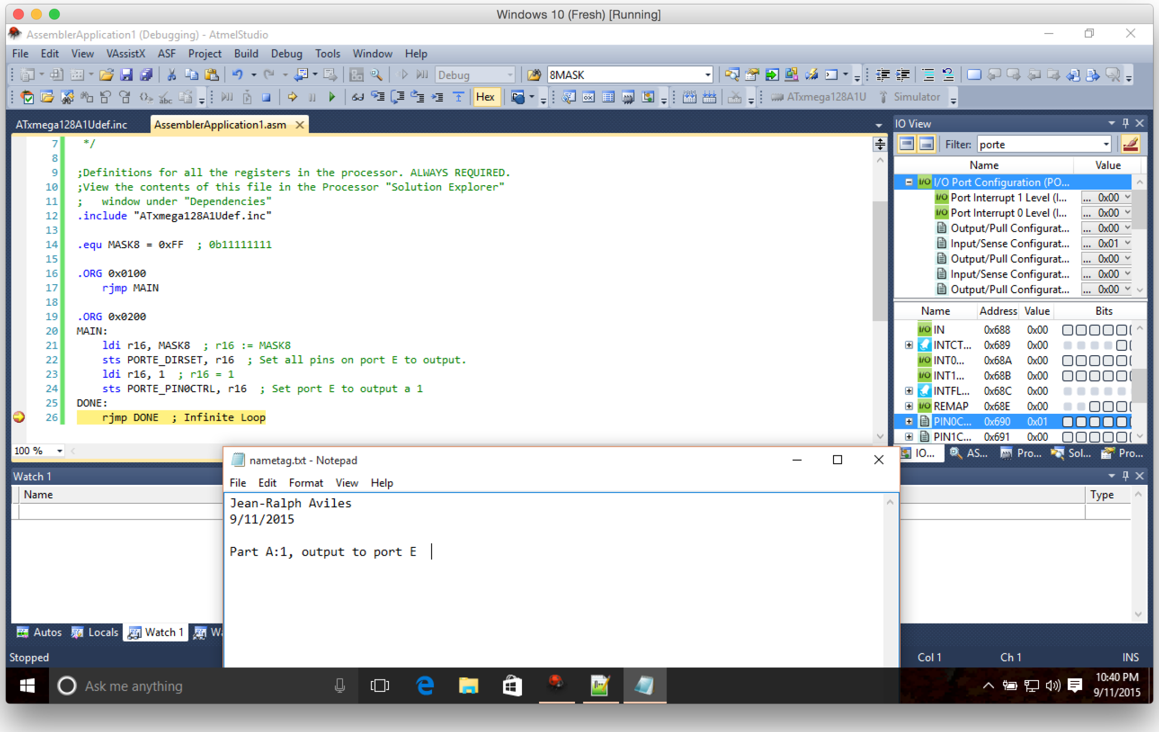
Future Work/Applications

I can now wire-wrap and I have a better idea of how to design and code for additions to the uP board.

Appendix

Part A

Program to write to Port E simulated


Windows 10 (Fresh) [Running]
AssemblerApplication1 (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
8MASK
Hex
ATxmega128A1U
Simulator
ATxmega128A1Udef.inc
AssemblerApplication1.asm
100 %
Watch 1
Name
Autos Locals Watch 1
Stopped
IO View
IO... AS... Pro... Sol... Pro...
Type
Col 1 Ch 1 INS
nametag.txt - Notepad
File Edit Format View Help
Ask me anything
10:40 PM
9/11/2015

```
 */

;Definitions for all the registers in the processor. ALWAYS REQUIRED.
;View the contents of this file in the Processor "Solution Explorer"
;   window under "Dependencies"
.include "ATxmega128A1Udef.inc"

.equ MASK8 = 0xFF  ; 0b11111111

.ORG 0x0100
    rjmp MAIN

.ORG 0x0200
MAIN:
    ldi r16, MASK8  ; r16 := MASK8
    sts PORTE_DIRSET, r16  ; Set all pins on port E to output.
    ldi r16, 1  ; r16 = 1
    sts PORTE_PIN0CTRL, r16  ; Set port E to output a 1
DONE:
    rjmp DONE  ; Infinite Loop
```

Filter: porte

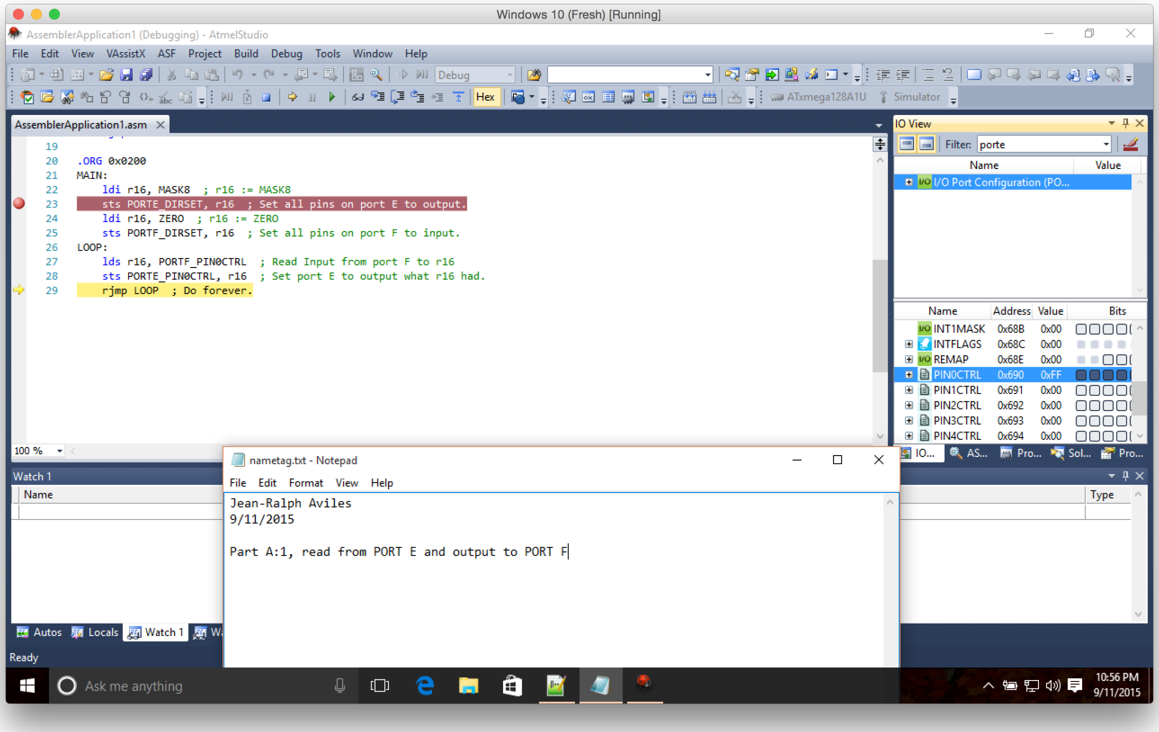
Name	Value
I/O Port Configuration (PO...	
Port Interrupt 1 Level (I...	0x00
Port Interrupt 0 Level (I...	0x00
Output/Pull Configurat...	0x00
Input/Sense Configurat...	0x01
Output/Pull Configurat...	0x00
Input/Sense Configurat...	0x00
Output/Pull Configurat...	0x00

Name	Address	Value	Bits
IN	0x688	0x00	□□□□□
INTCT...	0x689	0x00	□
INT0...	0x68A	0x00	□□□□□
INT1...	0x68B	0x00	□□□□□
INTFL...	0x68C	0x00	
REMAP	0x68E	0x00	□□□
PIN0C...	0x690	0x01	□□□□□
PIN1C...	0x691	0x00	□□□□□

```
Jean-Ralph Aviles
9/11/2015

Part A:1, output to port E
```

Program read from port F and write to Port E simulated



Windows 10 (Fresh) [Running]
AssemblerApplication1 (Debugging) - AtmelStudio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Debug
Hex
ATxmega128A1U
Simulator
AssemblerApplication1.asm
IO View
100 %
Watch 1
Name
Type
Autos Locals Watch 1
IO... AS... Pro... Sol... Pro...
Ready
nametag.txt - Notepad
File Edit Format View Help
Ask me anything
10:56 PM
9/11/2015

```
.ORG 0x0200
MAIN:
    ldi r16, MASK8  ; r16 := MASK8
    sts PORTE_DIRSET, r16  ; Set all pins on port E to output.
    ldi r16, ZERO  ; r16 := ZERO
    sts PORTF_DIRSET, r16  ; Set all pins on port F to input.
LOOP:
    lds r16, PORTF_PIN0CTRL  ; Read Input from port F to r16
    sts PORTE_PIN0CTRL, r16  ; Set port E to output what r16 had.
    rjmp LOOP  ; Do forever.
```

Filter: porte

Name	Value
I/O Port Configuration (PO...	

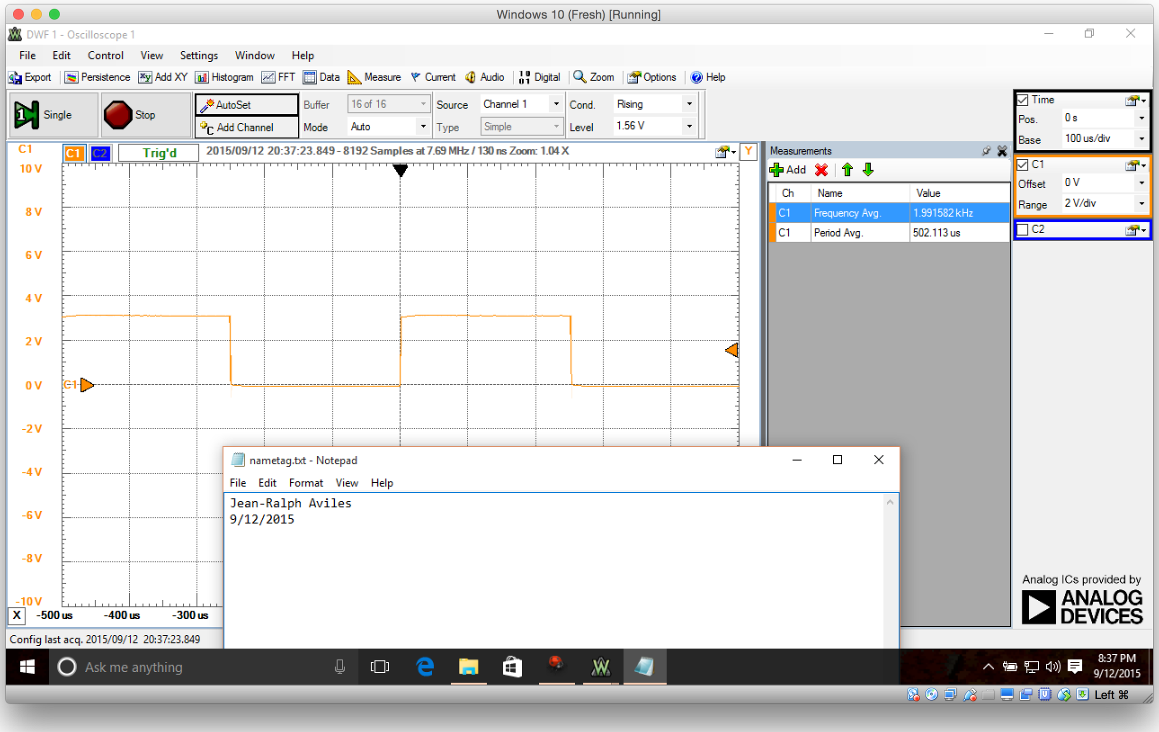
Name	Address	Value	Bits
INT1MASK	0x68B	0x00	□□□□
INTFLAGS	0x68C	0x00	
REMAP	0x68E	0x00	□□
PIN0CTRL	0x690	0xFF	■■■■
PIN1CTRL	0x691	0x00	□□□□
PIN2CTRL	0x692	0x00	□□□□
PIN3CTRL	0x693	0x00	□□□□
PIN4CTRL	0x694	0x00	□□□□

```
Jean-Ralph Aviles
9/11/2015

Part A:1, read from PORT E and output to PORT F
```

Part B

Program to Blink LED at 2kHz



Windows 10 (Fresh) [Running]
DWF 1 - Oscilloscope 1
File Edit Control View Settings Window Help
Export Persistence Add XY Histogram FFT Data Measure Current Audio Digital Zoom Options Help
Single
Stop
AutoSet
Add Channel
C1 C2 Trig'd
10 V
8 V
6 V
4 V
2 V
0 V
-2 V
-4 V
-6 V
-8 V
-10 V
-500 us
-400 us
-300 us
Measurements
Add
nametag.txt - Notepad
File Edit Format View Help
Ask me anything
Analog ICs provided by ANALOG DEVICES
8:37 PM
9/12/2015
Left ⌘

Buffer 16 of 16 · Mode Auto · Source Channel 1 · Type Simple · Cond. Rising · Level 1.56 V

Time ☑ · Pos. 0 s · Base 100 us/div

☑ C1 · Offset 0 V · Range 2 V/div

☐ C2

2015/09/12 20:37:23.849 - 8192 Samples at 7.69 MHz / 130 ns Zoom: 1.04 X

Ch	Name	Value
C1	Frequency Avg.	1.991582 kHz
C1	Period Avg.	502.113 us

```
Jean-Ralph Aviles
9/12/2015
```

Config last acq. 2015/09/12 20:37:23.849

Pseudocode/Flowcharts

Part A

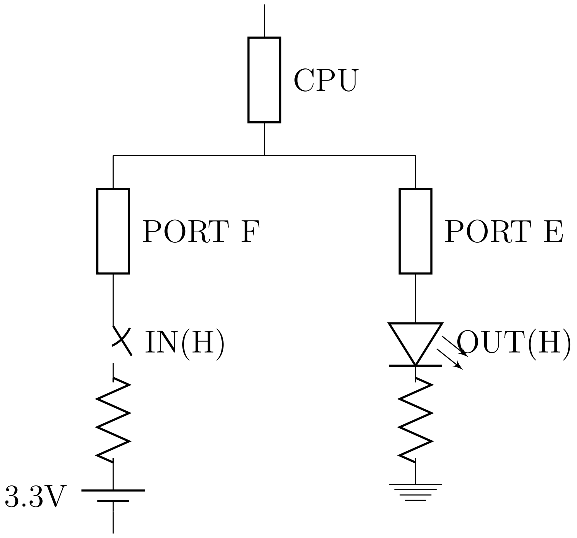
Program to write to Port E

```
BEGIN PROGRAM:
  CALL SET(PORT_E_1, HIGH)
END PROGRAM.
```

Program to read from Port F and read from Port E

```
BEGIN PROGRAM:
  X := READ(PORT_F_1)
  CALL SET(PORT_E_1, X)
END PROGRAM.
```

I/O Circuit Diagram



Part B

Program to Blink LED at 2kHz

```
BEGIN PROGRAM:
  WHILE TRUE
  DO
    LED_ON()
    SLEEP(250)
    LED_OFF()
    SLEEP(250)
  END WHILE
END PROGRAM.
```

Part C

Program for Part C

```
BEGIN PROGRAM:
  IF PORT_F_6 is clear
  THEN
    OUTPUT(PORT_E, 0x03)
    WHILE PORT_E_6 is clear
    DO
      DELAY(240ms)
      ROTATE_L(PORT_E, WRAP=TRUE)
    END WHILE
  ENDIF
  IF PORT_F_6 is not clear
  THEN
    OUTPUT(PORT_E, 0b11100111)
    * Simple algorithm for part C
    * E = 0xE7 and have a variable x := 0x24 and while
    * PORT F is not clear then...
    * xor(PORT_E, x)
    * swap nibles of X, so 0x24 => 0x42, 0x42 => 0x24
    * xor(PORT_E, x)
    * repeat
    x := 0x24
    DELAY(420ms)
    WHILE PORT_F_6 is not clear
      XOR(PORT_E, x)
      DELAY(420ms)
      IF PORT_F_6 is not clear
      THEN
        BREAK
      ENDIF
      SWAP_NIBBLES(x)
      XOR(PORT_E, x)
      DELAY(420ms)
    END WHILE
  ENDIF
END PROGRAM.
```

Programs

Part A

Program to write to Port E

```
/*
 * PartA_1.asm
 *
 *  Created: 9/11/2015
 *   Author: Jean-Ralph Aviles
 *  This program is an example program to output from port E.
 */

; Definitions for all the registers in the processor. ALWAYS
; REQUIRED. View the contents of this file in the Processor
; "Solution Explorer" window under "Dependencies"
.include "ATxmega128A1Udef.inc"
```

```
.equ MASK8 = 0xFF  ; 0b11111111

.ORG 0x0100
  rjmp MAIN

.ORG 0x0200
MAIN:
  ldi r16, MASK8  ; r16 := MASK8
  sts PORTE_DIRSET, r16  ; Set all pins on port E to output.
  ldi r16, 0xFF  ; r16 = 1
  sts PORTE_OUT, r16  ; Set port E to output
DONE:
  rjmp DONE  ; Infinite Loop
```

Program to read from Port F and read from Port E

```
/*
 * Lab2a_JA.asm
 *
 *  Created: 9/11/2015
 *   Author: Jean-Ralph Aviles
 *   Section: 1539
 *   TA: Khaled
 *  This program is an example program to read from port F
 *  and output to port E.
 */

; Definitions for all the registers in the processor. ALWAYS
; REQUIRED. View the contents of this file in the Processor
; "Solution Explorer" window under "Dependencies"
.include "ATxmega128A1Udef.inc"

.equ MASK8 = 0xFF  ; 0b11111111
.equ ZERO = 0x00  ; 0b00000000

.ORG 0x0100
  rjmp MAIN

.ORG 0x0200
MAIN:
  ldi r16, MASK8  ; r16 := MASK8
  sts PORTE_DIRSET, r16  ; Set all pins on port E to output.
  ldi r16, ZERO  ; r16 := ZERO
  sts PORTF_DIRSET, r16  ; Set all pins on port F to input.
LOOP:
  lds r16, PORTF_IN  ; Read Input from port F to r16
  sts PORTE_OUT, r16  ; Set port E to output what r16 had.
  rjmp LOOP  ; Do forever.
```

Part B

Program to Blink LED at 2kHz

```
/*
 * lab2b_JA.asm
 *
 *  Created: 9/11/2015
 *   Author: Jean-Ralph Aviles
 *   Section: 1539
 *   TA: Khaled
 * This program blinks the LEDs on port E at 2kHz.
 */

; Definitions for all the registers in the processor. ALWAYS
; REQUIRED. View the contents of this file in the Processor
; "Solution Explorer" window under "Dependencies"
.include "ATxmega128A1Udef.inc"

.equ MASK8 = 0xFF  ; 0b11111111
.equ NITERATIONS = 165

.ORG 0x0100
  rjmp MAIN

.ORG 0x0200
MAIN:
  ldi r16, MASK8  ; r16 := MASK8
  sts PORTE_DIRSET, r16  ; Set all pins on port E to output
LOOP:
  sts PORTE_OUTTGL, r16 ; Toggle OUTPUT
  CALL DELAY  ; Delay
  rjmp LOOP

DELAY:
  ldi r17, NITERATIONS ; Load r17 with delay counter
DELAYLOOP:
  dec r17 ; r17 = r17 - 1
  brne DELAYLOOP
  ret
```

Part C

```
/*
 * Lab2c_JA.asm
 *
 *  Created: 9/11/2015
 *   Author: Jean-Ralph Aviles
 *   Section: 1539
 *   TA: Khaled
 *  This program flashes some fancy animations depending on
 *  the status of PORT F.
 */

; Definitions for all the registers in the processor. ALWAYS
; REQUIRED. View the contents of this file in the Processor
; "Solution Explorer" window under "Dependencies"
.include "ATxmega128A1Udef.inc"

.org 0x0100
  rjmp MAIN

.org 0x200
MAIN:
  ldi r16, 0x00
  sts PORTF_DIRSET, r16  ; Set PORTF to input
  ldi r16, 0xFF
  sts PORTE_DIRSET, r16  ; Set PORTE to output
  call GET_PORT_6  ; r16 = PORT_F_6
  cpi r16, 0x0  ; compare PORT_F_6
  brne B  ; If PORT_F_6 is set goto C
  breq A  ; Else go to A

A:
  ldi r16, 0x03  ; r16 = 0x00000011
A_1:
  call SET_OUTPUT  ; Output r16
  push r16  ; Save r16
  ldi r16, 24 ; r16 = 24
  call DELAY  ; Delay for 24*10 = 240ms
  call GET_PORT_6  ; r16 = PORT_F_6
  brne A_EXIT ; Break if PORT_F_6 is not zero
  pop r16  ; Restore r16
  rol r16  ; Rotate left
  brcc A_2; If there was no carry bit
  ori r16, 0x01  ; Add bit that "fell off"
A_2:
  rjmp A_1  ; Loop
A_EXIT:
  pop r16  ; Make sure to clean the stack
  rjmp B  ; Go to other Pattern

B:
  ldi r17, 0x24  ; Load r17 with 0x24
  ldi r16, 0xE7  ; Load r16 with 0b11100111
B_1:
  call SET_OUTPUT  ; Output r16
  push r16  ; Save r16
  ldi r16, 42  ; r16 = 42
  call DELAY  ; Delay for 42*10 = 420ms
  call GET_PORT_6  ; r16 = PORT_F_6
  breq B_EXIT  ; Break if PORT_F_6 is zero
  pop r16  ; Restore r16
  eor r16, r17  ; r16 = r16 ^ r17
  swap r17  ; Swap nibbles of r17

  call SET_OUTPUT  ; Output r16
  push r16  ; Save r16
  ldi r16, 42  ; r16 = 42
  call DELAY  ; Delay for 42*10 = 420ms
  call GET_PORT_6  ; r16 = PORT_F_6
  breq B_EXIT  ; Break if PORT_F_6 is zero
  pop r16  ; Restore r16
  eor r16, r17  ; r16 = r16 ^ r17
  rjmp B_1  ; Loop
B_EXIT:
  pop r16  ; Make sure to clean the stack
  rjmp A  ; Go to other Pattern


GET_PORT_6: ; Returns port 6 into r16
  lds r16, PORTF_IN  ; Get PORTF input
  andi r16, 0x40  ; We only care about bit 6
  ret

SET_OUTPUT: ; Outputs r16 to Port E
   sts PORTE_OUT, r16  ; Set output to r16
   ret

DELAY: ; Delays by r16 x 10ms
  ; 66 * 100 instructions is ~ 10ms
  push r16  ; Push r16 onto the stack
  cpi r16, 0  ; Compare counter
  breq DELAY_RET  ; If counter is 0 return
DELAY_LOOP:
  push r16  ; Push counter onto the stack
  ldi r16, 66; Load outer time loop
DELAY_OUTERLOOP:
  push r16  ; Push outer time loop onto the stack
  ldi r16, 100 ; Load inner time loop
DELAY_INNERLOOP:
  dec r16  ; Decrement inner counter
  brne DELAY_INNERLOOP
  pop r16  ; Pop outer time loop off the stacl
  dec r16  ; Decrement outer counter
  brne DELAY_OUTERLOOP
  pop r16  ; Pop counter off of stack
  dec r16  ; Decrement Counter
  brne DELAY_LOOP  ; If counter is not zero loop again
DELAY_RET:
  pop r16 ; Pop original r16 off of the stack
  ret
```