Lab 5

Michael Arboleda
Lab Section: 7F34

July 12, 2017

b. Answers to all pre-lab questions

1) What is the difference in conversion ranges between 8-bit unsigned and signed conversion modes? List both ranges.

ANS: Conversion Range for unsigned 8-bit: 0 to 255. Conversion for signed 8-bit: -128 to 127. While both ranges represent 256 numbers, unsigned can use larger numbers but not negative numbers.

2) Assume you wanted a voltage reference range from -2 V to 3 V, with an unsigned 12-bit ADC. What are the voltages if the ADC output is 0xA92 and 0x976?

ANS: The voltage for 0xA92 is 1.304 V. The voltage for 0x976 is 0.957 V.

3) Write the address decode equation to put the input and output ports at addresses 0x2000 0xAFFF.

ANS: $(\overline{A15} * \overline{A14} * A13) + (\overline{A15} * A14) + (A15 * \overline{A14} * \overline{A13}) + (A15 * \overline{A14} * A13 * \overline{A12})$;

c. Problems Encountered

"Problems" encounter came from the lab doc not being clear enough. AS i did the lab, I assumed thing based on the language, however, TAs clarified up the confusion. Examples include that freerunner was not supposed to be use, so I had to fix my part b,c,d,e. Also the menu choices because I thought the lab wanted a continous display of other CdS or te DAD, however only one was needed at a time,

d. Future Work/Applications

In this lab only one configuration for ADC was used. It would be useful to try and make working programs for different ADC configurations, including Free Run and other gain styles. Also PWM can be used with the LED to change the dimness of the LED in relation to the value red by the CdS.

e. Schematics

N/A

g. Pseudocode/Flowcharts

Pseudocode for Lab5a.c:

```
MAIN:
    * Call Change_CLK_32HZ Function

WHILE(TRUE){
* Store 0x37 in 0x8500
* Store 0x73 in 0x8501
}
END


FUNCTION EBI_INIT
    * Set up EBI CTRL
    * Set up EBI CHIP 0 and base address
    * Set up PORT H
    * Set up PORT K


FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

Pseudocode for Lab5b.c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call ADC_INIT() Function

WHILE(TRUE){}
    * Store ADC value
END

FUNCTION ADC_INIT
    * SET PORT A as input
    * SET up Control A and B
    * Set up REf and Pre scaler controls
    * Set up Chan 0 ctrl/mux ctrl

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

Pseudocode for Lab5c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call ADC_INIT() Function
    * Call USART_INIT() Function

WHILE(TRUE){}
    * Store ADC value
END

FUNCTION int_to_char
    * int x;
    switch(x)
        case x: return 'x'

FUNCTION volt_string
    IF(x is negative){
        * y = 2s compliment of x
    }
    * Store first digit of f1
    * Bring digit 2 to front
    * Store 2nd digit of f1
    * Store 3rd digit of f1
    * Make string with volt output
    * Store string in char array passed in

FUNCTION USART_INIT
    * Set port D for USART com
    * Set up Ctrl B and C
    * Set up baud rate

FUNCTION OUT_STRING
    While(char[] does not equal NULL){
        * Call OUT_CHAR
    }

FUNCTION OUT_CHAR
    While(Transmitting){}
    * Return USART Data in a CHAR
```

```
FUNCTION ADC_INIT
    * SET PORT A as input
    * SET up Control A and B
    * Set up REf and Pre scaler controls
    * Set up Chan 0 ctrl/mux ctrl

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

Pseudocode for Lab5d.c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call ADC_INIT() Function
    * Call USART_INIT() Function

WHILE(TRUE){}
    * Store ADC value
END

FUNCTION int_to_char
    * int x;
    switch(x)
        case x: return 'x'

FUNCTION volt_string
    IF(x is negative){
        * y = 2s compliment of x
    }
    * Store first digit of f1
    * Bring digit 2 to front
    * Store 2nd digit of f1
    * Store 3rd digit of f1
    * Make string with volt output
    * Store string in char array passed in

FUNCTION USART_INIT
    * Set port D for USART com
    * Set up Ctrl B and C
    * Set up baud rate

FUNCTION OUT_STRING
    While(char[] does not equal NULL){
        * Call OUT_CHAR
    }

FUNCTION OUT_CHAR
    While(Transmitting){}
    * Return USART Data in a CHAR
```

```
FUNCTION ADC_INIT
    * SET PORT A as input
    * SET up Control A and B
    * Set up REf and Pre scaler controls
    * Set up Chan 1 ctrl/mux ctrl

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

Pseudocode for Lab5e.c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call ADC_INIT() Function
    * Call USART_INIT() Function
    * Call COUNTER_INIT() Funcion
    * Call EBI_INIT() Function
    WHILE(TRUE){
     * Take user char
        switch(char){
            case A:
            case a:
                * Start conversion on CdS Cell
                * Display on terminal
         case B:
         case b:
                * Start conversion on Dad
                * Display on terminal
         case C:
         case c:
                * Start counter
         case D:
         case d:
                * Stop counter
         case E:
         case e:
                * Start conversion on CdS Cell
                * Store in SRAM
         case F:
         case f:
                * Read from SRAM
                * Display on terminal
        }
    }
END

FUNCTION COUNTER_INIT
    * SET top of counter

FUNCTION COUNTER_START
    * SET interupt for counter
    * Start COUNTER

FUNCTION COUNTER_STOP
    * STOP TIMER INTERRUPT
    * STOP TIMER
    * RESET TIMER
```

```
ISR(TCC0_OVF_vect)
    * Start conversion on CdS Cell
    * Display on terminal

FUNCTION display_menu
* Set up Menu String pointer
* Set up array of strings pointers
* Loop thru menu array to display

FUNCTION int_to_char
    * int x;
    switch(x)
        case x: return 'x'

FUNCTION volt_string
    IF(x is negative){
        * y = 2s compliment of x
    }
    * Store first digit of f1
    * Bring digit 2 to front
    * Store 2nd digit of f1
    * Store 3rd digit of f1
    * Make string with volt output
    * Store string in char array passed in

FUNCTION USART_INIT
    * Set port D for USART com
    * Set up Ctrl B and C
    * Set up baud rate

FUNCTION OUT_STRING
    While(char[] does not equal NULL){
        * Call OUT_CHAR
    }

FUNCTION OUT_CHAR
    While(Transmitting){}
    * Return USART Data in a CHAR

FUNCTION ADC_INIT
    * SET PORT A as input
    * SET up Control A and B
    * Set up REf and Pre scaler controls
    * Set up Chan 0 ctrl/mux ctrl/INT
    * Set up Chan 1 ctrl/mux ctrl/INT
* Set up compare value for interrupt
```

```
ISR(ADCA_CH0_vect)
    * TURN OFF all lights
    * TURN ON RED
* Clear Flag

ISR(ADCA_CH1_vect)
* TURN OFF all lights
* TURN ON BLUE
* Clear Flag

FUNCTION EBI_INIT
    * Set up EBI CTRL
    * Set up EBI CHIP 0 and base address
    * Set up PORT H
    * Set up PORT K

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

h. Program Code

Code for Lab5a.c:

```
/*Lab 5 Part A

Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Uses EBI to store value

Lab5a.c
Created: 7/7/2017 11:11:20 PM

*/

#include <avr/io.h>
#include "constants.h"
#include "Clk_32MHz.h"
#include "USART.h"
#include "EBI.h"


int main(void){
   //Set up program
   Change_CLK_32HZ();
   USART_INIT();
   EBI_INIT();

   //Set up Pointer at address
   uint8_t *ptr1 = 0x8500;
   uint8_t *ptr2 = 0x8501;

   //Store values at mem location
   while(TRUE){
      *ptr1 = 0x37;
      *ptr2 = 0x73;
   }
}
```

Code for Lab5b.c:

```
/*
Lab 5 Part B
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description:

Lab5b.c
Created: 7/10/2017 4:16:07 AM
 */

#include <avr/io.h>
#include "Clk_32MHz.h"
#include "constants.h"
#include "ADC.h"

int main(void){
    //Set up program
   Change_CLK_32HZ();
   ADC_INIT();

   uint8_t x;

   //While 1
   while(TRUE){
      // Read CdS and store in x
      ADCA.CH0.CTRL |= PIN7_bm;
      x = ADCA.CH0.RESL;
   }
}
```

Code for Lab5c.c

```c
/*
Lab 5 Part C
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Voltmeter

Lab5c.c
Created: 7/10/2017 4:44:33 AM
 */

#include <avr/io.h>
#include "Clk_32MHz.h"
#include "USART.h"
#include "constants.h"
#include "ADC.h"
#include "ATXMEGA_DISPLAY.h"

int main(void){
   // Set up program
   Change_CLK_32HZ();
   USART_INIT();
   ADC_INIT();


   char c[15];
   uint8_t x;

   //While 1
   while(TRUE){
      //Read CdS and store in x
      ADCA.CH0.CTRL |= PIN7_bm;
      x = ADCA.CH0.RESL;

      //Display on terminal
      volt_string(c,x);
      OUT_STRING(c);
   }
}
```

Code for Lab5d.c

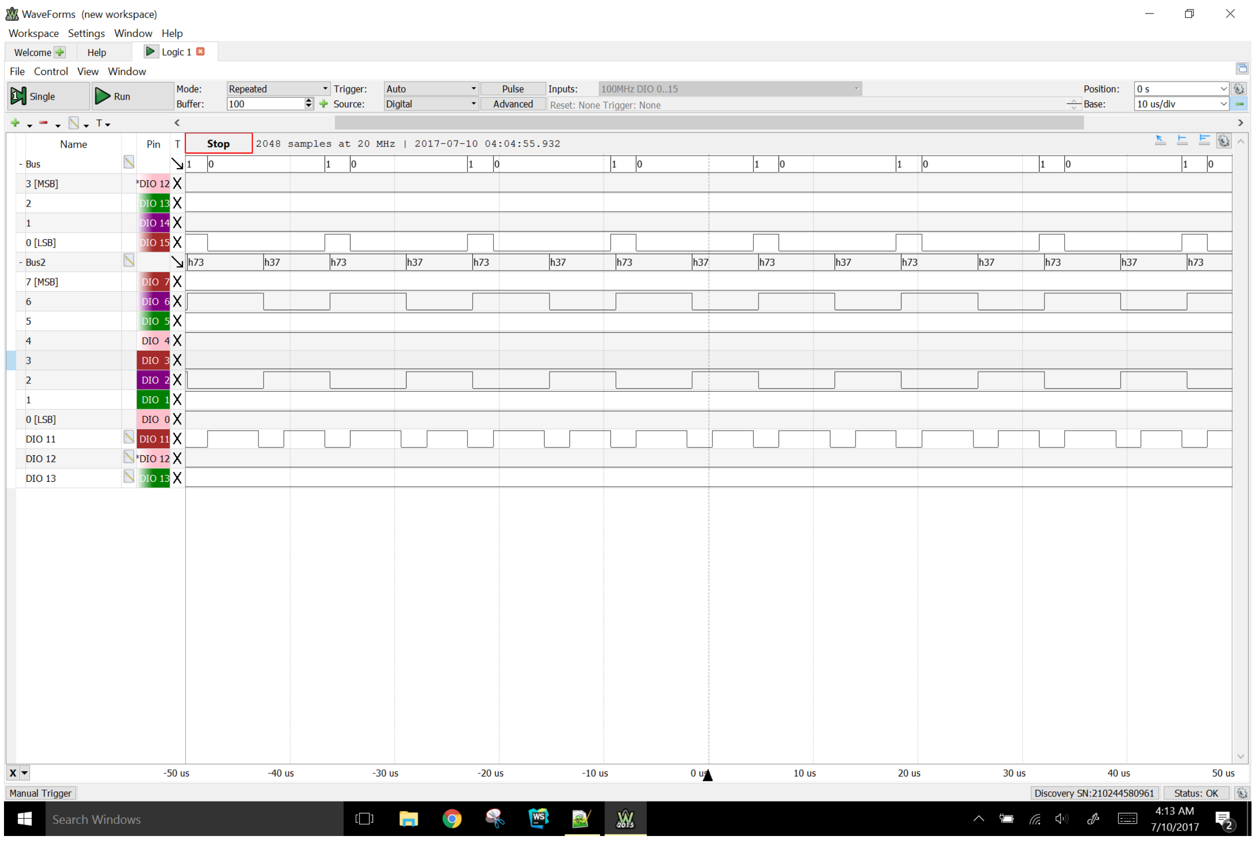
```c
/*Lab 5 Part D
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: ADC with DAD

Lab5d.c
Created: 7/12/2017 2:46:07 AM
*/

#include <avr/io.h>
#include "Clk_32MHz.h"
#include "USART.h"
#include "constants.h"
#include "ADC.h"
#include "ATXMEGA_DISPLAY.h"

int main(void){
   //Set up program
   Change_CLK_32HZ();
   USART_INIT();
   ADC_INIT();

   char c[15];
   uint8_t x;

   while(TRUE){
      // Read DAD data
      ADCA.CH1.CTRL |= PIN7_bm;
      x = ADCA.CH1.RESL;


      volt_string(c,x);
      OUT_STRING(c);
   }
}
```

Code for Lab5e.c

```
/*Lab 5 Part E
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: EBI, and ADC


Lab5e.c
Created: 7/11/2017 10:48:13 AM
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include "Clk_32MHz.h"
#include "constants.h"
#include "ADC.h"
#include "USART.h"
#include "ATXMEGA_DISPLAY.h"
#include "TIMER_COUNTER.h"
#include "EBI.h"

//Prototype
void display_menu(void);

int main(void){
   //Set up program
   Change_CLK_32HZ();
   USART_INIT();
   ADC_INIT();
   COUNTER_INIT();
   EBI_INIT();

   //Display Menu
   display_menu();

   //Set up variables
   char c_in;
   char c[15];
   volatile uint8_t x;
   volatile uint8_t *ptr1 = 0x8000;

   //TURN ON LED
   PORTD_DIRSET = (PIN4_bm | PIN6_bm);
   PORTD_OUTSET = 0x50;

   //Set up interrupts
   uint8_t PMIC_crtl_lvl_config = 0b00000111;
   PMIC_CTRL = PMIC_crtl_lvl_config;
   sei();
```

```
while(TRUE){
   //Take user input
   c_in = IN_CHAR();

   //Switch on choice
   switch(c_in){
      //IF a
      case 'a':
      case 'A':
         //Start conversion on CdS Cell
         ADCA.CH0.CTRL |= PIN7_bm;
         x = ADCA.CH0.RESL;

         //Display on terminal
         volt_string(c,x);
         OUT_STRING(c);
         break;
      //IF b
      case 'b':
      case 'B':
         //Start conversion on PINs for ch1
         ADCA.CH1.CTRL |= PIN7_bm;
         x = ADCA.CH1.RESL;

         //Display on terminal
         volt_string(c,x);
         OUT_STRING(c);

         break;
      //If c
      case 'c':
      case 'C':
         //Start counter
         COUNTER_START();
         break;
      //IF d
      case 'd':
      case 'D':
         //Stop counter
         COUNTER_STOP();
         break;
      //IF e
      case 'e':
      case 'E':
         //Get CdS ande store in SRAM
         ADCA.CH0.CTRL |= PIN7_bm;
         x = ADCA.CH0.RESL;
         *ptr1 = x;
         break;
      case 'f':
         case 'F':
            //Take SRAM vulue
            x = *ptr1;

            //Display on terminal
            volt_string(c,x);
            OUT_STRING(c);
            break;
      }
   }
   return 0;
}

void display_menu(void){
   uint8_t menu_size = 6;

   //Menu String
   char *menuA = "A/a:␣Start/Display␣conversion␣on␣CdS␣Cell\r\n";
   char *menuB = "B/b:␣Start/Display␣conversion␣on␣DAD/NAD␣channel\r\n";
   char *menuC = "C/c:␣1␣second␣timer␣CdS\r\n";
   char *menuD = "D/d:␣Turn␣off␣timer\r\n";
   char *menuE = "E/e:␣Store␣CdS␣in␣SRAM\r\n";
   char *menuF = "F/f:␣Read/Display␣SRAM␣data\r\n";

   //Set up array of menu strings
   char *menu[] = {menuA, menuB, menuC, menuD, menuE, menuF};

   //Loop thru menu array to display
   for(uint8_t i = 0; i < menu_size; i++){
      OUT_STRING(menu[i]);
   }

   //return
   return;
}
```

i. Appendix


Mode: Repeated
Buffer: 100
Trigger: Auto
Source: Digital
Inputs: 100MHz DIO 0..15
Reset: None Trigger: None
Position: 0 s
Base: 10 us/div

```
2048 samples at 20 MHz | 2017-07-10 04:04:55.932
```

Discovery SN:210244580961
Status: OK

Figure 1: DAD reading of EBI

0x37 is written to address 0x8500 when WE and CS0 are both true/active low. ALE1 latches to higher byte of address, which should happen since it is time-mutiplexed. 0x73 is written to address 0x8501 when WE and CS0 are both true/qctive low. ALE1 latches to higher byte of address, as stated above.

EQUATION:

Using 2 points (0x17, 0.45 V) and (0x54, 1.67 V) obtained from the voltmeter and register we can find the slope:

$$\frac{1.67 - 0.45}{0x54 - 0x17} = \frac{1.22}{0x3D} = 0.02$$

Code for ADC.h:

```
#ifndef ADC_H_
#define ADC_H_
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Sets up ADC

ADC.h
Created: 7/11/2017 2:50:03 AM
*/

//INCLUDES
#include <avr/io.h>
#include <avr/interrupt.h>
#include "constants.h"

//IMPORT CONSTANTS
extern const uint8_t ADC_CTRLA_CONFIG;
extern const uint8_t ADC_CTRLB_CONFIG;
extern const uint8_t ADC_REFCTRL_CONFIG;
extern const uint8_t ADC_PRESCALER_CONFIG;
extern const uint8_t ADC_CH0_CTRL_CONFIG;
extern const uint8_t ADC_CH0_MUXCTRL_CONFIG;
extern const uint8_t ADC_PORTA;
extern const uint8_t ADC_CH1_CTRL_CONFIG;
extern const uint8_t ADC_CH1_MUXCTRL_CONFIG;
extern const uint8_t ADC_CH0_INTCTRL_CONFIG;
extern const uint8_t ADC_CH1_INTCTRL_CONFIG;
extern const uint8_t ADC_CMP_CONFIG;

void ADC_INIT(){
   //SET PORT A as input
   PORTA_DIRCLR = ADC_PORTA;

   //SET up Control A and B
   ADCA.CTRLA = ADC_CTRLA_CONFIG;
   ADCA.CTRLB = ADC_CTRLB_CONFIG;

   // Set up REf and Pre scaler controls
   ADCA.REFCTRL = ADC_REFCTRL_CONFIG;
   ADCA.PRESCALER = ADC_PRESCALER_CONFIG;

   // Set up Chan 0 ctrl/mux ctrl
   ADCA.CH0.CTRL = ADC_CH0_CTRL_CONFIG;
   ADCA.CH0.MUXCTRL = ADC_CH0_MUXCTRL_CONFIG;
   ADCA.CH0.INTCTRL = ADC_CH0_INTCTRL_CONFIG;

   // Set up Chan 1 ctrl/mux ctrl
   ADCA.CH1.CTRL = ADC_CH1_CTRL_CONFIG;
   ADCA.CH1.MUXCTRL = ADC_CH1_MUXCTRL_CONFIG;
   ADCA.CH1.INTCTRL = ADC_CH1_INTCTRL_CONFIG;

   //Set up compare value for interrupt
   ADCA.CMP = ADC_CMP_CONFIG;
}


ISR(ADCA_CH0_vect){
   //TURN OFF all lights
   PORTD_OUTSET = 0x50;

   //TURN ON RED
   PORTD_OUTCLR = PIN4_bm;

   //CLear Flag
   ADCA.CH0.INTFLAGS = 0x01;

   return;
}

ISR(ADCA_CH1_vect){
   //TURN OFF all lights
   PORTD_OUTSET = 0x50;

   //TURN ON BLUE
   PORTD_OUTCLR = PIN6_bm;

   //CLear Flag
   ADCA.CH1.INTFLAGS = 0x01;

   return;
}

#endif /* ADC_H_ */
```

Code for ATXMEGA_DISPLAY.h:

```
#ifndef ATXMEGA_DISPLAY_H_
#define ATXMEGA_DISPLAY_H_
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Sets up Chars
        and strings for USART

ATXMEGA_DISPLAY.h
Created: 7/11/2017 6:22:24 AM
*/

#include <avr/io.h>


/********************Function************************************
Function Name: int_to_char
Inputs: uint8_t x
Outputs: No outputs
*/
char int_to_char(uint8_t x){

   // Return char verion of hex digit
   switch(x){
      case 0: return '0';
      case 1: return '1';
      case 2: return '2';
      case 3: return '3';
      case 4: return '4';
      case 5: return '5';
      case 6: return '6';
      case 7: return '7';
      case 8: return '8';
      case 9: return '9';
      case 10: return 'A';
      case 11: return 'B';
      case 12: return 'C';
      case 13: return 'D';
      case 14: return 'E';
      case 15: return 'F';
   }
   // Return NULL if invalid number
   return NULL;
}



/********************Function************************************
Function Name: volt_string
```

```
Inputs: char * c, uint8_t x
Outputs: No outputs
*/
void volt_string(char * c, uint8_t x){

   // Change to decimal form of signed num
   float f1;
   uint8_t y = x;

   if((x&PIN7_bm) == PIN7_bm){
      y = ((~(x))+1);
   }
   f1 = ((y*1.0)*0.02);



   //Store first digit of f1
   uint8_t d1 = (uint8_t)((int)f1);

   //Bring digit 2 to front
   float f2 = 10.0*(f1-d1);

   //Store 2nd digit of f1
   uint8_t d2 = (uint8_t) ((int)f2);

   //Store 3rd digit of f1
   uint8_t d3 = (uint8_t) ((int) (10.0*(f2-d2)));

   //Make string with volt output
   char ca[] = { ((x & PIN7_bm) == PIN7_bm ? '-': '+'), int_to_char(d1),
      '.', int_to_char(d2), int_to_char(d3), ' ', 'V', ' ', '(',
      int_to_char((x >> 4) & 0x0F) ,int_to_char(x & 0x0F), ')', '\r', '\n',NULL
   };

   //Store string in char array passed in
   for(int i = 0; i < 15; i++){
      //store each char in order
      c[i] = ca[i];
   }
   //return
   return;
}




#endif /* ATXMEGA_DISPLAY_H_ */
```

Code for Clk_32MHz.h:

```
#ifndef CLK_32MHZ_H_
#define CLK_32MHZ_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Changes uP freq to 32Mhz

Clk_32MHz.h
Created: 7/7/2017 11:20:13 PM
*/
#include <avr/io.h>
#include "constants.h"

//include extern constants
extern const uint8_t NEW_CLOCK_FREQ;


/********************Function*************************************
; Function Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
*/
void Change_CLK_32HZ(){

   //Set the clk config
   OSC_CTRL = NEW_CLOCK_FREQ;

   //Wait for the right flag to be set in the OSC_STATUS reg
   while((OSC_STATUS & PIN1_bm) != PIN1_bm);

   //Write the IOREG signature to the CPU_CCP reg
   CPU_CCP = CCP_IOREG_gc;

   //Select the new clock source in the CLK_CTRL reg
   CLK_CTRL = CLK_SCLKSEL_RC32M_gc;

   return;
}

#endif /* CLK_32MHZ_H_ */
```

Code for EBI.h:

```c
#ifndef EBI_H_
#define EBI_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Set up EBI

EBI.h
Created: 7/10/2017 1:20:14 AM
 */
extern const uint8_t ebi_ctrl_config;
extern const uint8_t ebi_baseaddlL_config;
extern const uint8_t ebi_baseaddlH_config;
extern const uint8_t PORTH_CONFIG;
extern const uint8_t PORTK_CONFIG;


void EBI_INIT(void){
   // Set up EBI CTRL
   EBI.CTRL = ebi_ctrl_config;

   //Set up EBI CHIP 0 and base address
   EBI.CS0.CTRLA = ebi_ctrla_config;
   EBI.CS0.BASEADDRH = ebi_baseaddlH_config;
   EBI.CS0.BASEADDRL = ebi_baseaddlL_config;

   //Set up PORT H
   PORTH_DIRSET = PORTH_CONFIG;
   PORTH_OUTSET = PORTH_CONFIG;

   //Set up PORT K
   PORTK_DIRSET = PORTK_CONFIG;

}




#endif /* EBI_H_ */
```

Code for TIMER_COUNTER.h:

```
#ifndef TIMER_COUNTER_H_
#define TIMER_COUNTER_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Timer Counter

TIMER_COUNTER.h
Created: 7/12/2017 6:09:50 AM
*/
#include <avr/io.h>
#include <avr/interrupt.h>
#include "constants.h"
#include "ATXMEGA_DISPLAY.h"

extern const uint16_t TC_PER_CONFIG;
extern const uint8_t INTCTRLA_CONFIG;
extern const uint8_t CLK_DIV_CONFIG;

void COUNTER_INIT(void){
   // SET top of counter
   TCC0.PER = TC_PER_CONFIG;

   return;
}

void COUNTER_START(void){
   //SET interupt for counter
   TCC0.INTCTRLA = INTCTRLA_CONFIG;

   //Start COUNTER
   TCC0_CTRLA = CLK_DIV_CONFIG;

   return;
}

void COUNTER_STOP(void){

   //STOP TIMER INTERRUPT
   TCC0.INTCTRLA = 0x00;

   //STOP TIMER
   TCC0.CTRLA = 0x00;

   //RESET TIMER
   TCC0_CTRLFCLR = TC_CMD_RESTART_gc;

   return;
}


ISR(TCC0_OVF_vect){
   char c[15];

   ADCA.CH0.CTRL |= PIN7_bm;
   uint8_t x = ADCA.CH0.RESL;

   //Display on terminal
   volt_string(c,x);
   OUT_STRING("ISR:␣");
   OUT_STRING(c);
}



#endif /* TIMER_COUNTER_H_ */
```

Code for USART.h:

```
#ifndef USART_H_
#define USART_H_
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Set up USART

USART.h
Created: 7/8/2017 3:07:25 AM
 */

#include <avr/io.h>
#include "constants.h"

extern const uint8_t pin_Tx;
extern const uint8_t pin_Rx;
extern const uint8_t TxRx_On;
extern const uint8_t usart_ctrl_C;
extern const uint8_t BSCALE;
extern const uint8_t upper_BSEL;
extern const uint8_t BSEL;
/********************Function*************************************
; Function Name: USART_INIT
; Inputs: No inputs
; Outputs: No outputs
*/
void USART_INIT(void){
   // Set port D for USART com
   PORTD.DIRSET = pin_Tx;
   PORTD.OUTSET = pin_Tx;
   PORTD.DIRCLR = pin_Rx;

   //Set up Ctrl B and C
   USARTD0_CTRLB = TxRx_On;
   USARTD0_CTRLC = usart_ctrl_C;

   //Set up baud rate
   USARTD0.BAUDCTRLA = BSEL;
   USARTD0.BAUDCTRLB = (BSCALE | upper_BSEL);

   return;
}



/********************Function*************************************
; Function Name: OUT_CHAR
; Inputs: Char c
; Outputs: No outputs
```

```
*/
void OUT_CHAR(char c){

   // Wait until prev receive done
   while((USARTD0.STATUS & PIN5_bm) == 0);

   //Output char thru USART
   USARTD0.DATA = c;

   return;
}



/********************Function*************************************
; Function Name: OUT_STRING
; Inputs: Char pointer c
; Outputs: No outputs
*/
void OUT_STRING(char* str){
   //While char is not null
   while(*str){
      //Output char
      OUT_CHAR(*str++);
   }
   return;
}



/********************Function*************************************
; Function Name: IN_CHAR
; Inputs: Char c
; Outputs: No outputs
*/
char IN_CHAR(void){
   while((USARTD0.STATUS & PIN7_bm) == 0);
   return USARTD0_DATA;
}


#endif /* USART_H_ */
```

Code for constants.h:

```
#ifndef CONSTANTS_H_
#define CONSTANTS_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Const declarations

constants.h
Created: 7/7/2017 11:53:20 PM
*/
#include <avr/io.h>

//OVERALL DEFS
#define TRUE 1
#define FALSE 0
#define NULL 0

//****************** CLK_32MHz.h ***********************************
const uint8_t NEW_CLOCK_FREQ = 0b00000010;



//****************** USART.h ***************************************
const uint8_t pin_Tx = PIN3_bm;
const uint8_t pin_Rx = PIN2_bm;
const uint8_t TxRx_On = 0b00011000; // 0x18
// asynch, 8 databits, no parity, 1 start, and 1 stop
const uint8_t usart_ctrl_C = 0b00000011;
// BSEL 28,800
const uint8_t upper_BSEL = 0b00001000;
const uint8_t BSEL = 0b10001110;
const uint8_t BSCALE = 0b10110000; // -5 BSCALE



//****************** EBI.h ***************************************
const uint8_t ebi_ctrl_config = 0b00000001;
const uint8_t ebi_ctrla_config = 0b00010101;
const uint8_t ebi_baseaddlL_config = 0x80;
const uint8_t ebi_baseaddlH_config = 0b00000000;
const uint8_t PORTH_CONFIG = 0b00010111;
const uint8_t PORTK_CONFIG = 0b11111111;



//****************** ADC.h ***************************************
const uint8_t ADC_CTRLA_CONFIG = 0b00000001; //0x01
const uint8_t ADC_CTRLB_CONFIG = 0b00010100; //0x14
const uint8_t ADC_REFCTRL_CONFIG = 0b00110000; //0x30
const uint8_t ADC_PRESCALER_CONFIG = 0b00000000;//0x07
const uint8_t ADC_CH0_CTRL_CONFIG = 0b00000011;  //0x03
const uint8_t ADC_CH0_MUXCTRL_CONFIG = 0b00001010; //0x0A
const uint8_t ADC_PORTA = 0b01000011;      //0x
//Channel 1
const uint8_t ADC_CH1_CTRL_CONFIG = 0b00000011;  //0x03
const uint8_t ADC_CH1_MUXCTRL_CONFIG = 0b00100001; //0x21
//^ Pin 4, PIN 5
//interrupts
const uint8_t ADC_CH0_INTCTRL_CONFIG = 0b00000111; //0x07
const uint8_t ADC_CH1_INTCTRL_CONFIG = 0b00001111; //0x0F
const uint8_t ADC_CMP_CONFIG = 0x00;  //0x00

//****************** TIMER_COUNTER.h ************************************
const uint16_t TC_PER_CONFIG = 0x8000;
const uint8_t INTCTRLA_CONFIG = 0b00000011;
const uint8_t CLK_DIV_CONFIG = 0b00000111;




#endif /* CONSTANTS_H_ */
```