HW3
Union College
ECE352
Assignment 4 Solution

1. The ADC is to be configured to measure an input voltage which ranges from 0-0.3 volts. Give the registers which must be configured and the values which you will load into them to give an accurate measurement of voltages in this range. This can be done in C or assembler.

There are many possible ways to configure the ADC and get a good measurement of the range. Here is what I chose for the configuration and the sequence of steps is outlined.

Enable ADC0:
```
mov REF0CN, #002h     ; Reference Control Register
mov ADC0CN, #080h     ; ADC Control Register
```

Single Ended Inputs:
```
mov AMX0CF, #000h     ; AMUX Configuration Register
mov AMX0SL, #000h     ; AMUX Channel Select Register – assume channel 0
```

Clock is Default and Gain is 1
```
mov ADC0CF, #0F8h     ; ADC0 Configuration Register
```

ADC0 Reference Configuration – use DAC0 Output
```
mov REF0CN, #012h     ; Reference Control Register
```

Also, internal voltage reference for DAC0
```
mov REF0CN, #013h     ; Reference Control Register
```

DAC0 Configuration – use 0.3V as the reference voltage
```
mov DAC0CN, #080h     ; DAC0 Control Register
mov DAC0L, #000h              ; DAC0 Low Byte Register
mov DAC0H, #002h              ; DAC0 High Byte Register
```

With this configuration, the swing from 0-0.3V will result in a value from 000-FFF in the ADC registers.

It is also possible to use the gain to fully utilize the range of digital values.

2. The 8051 is configured to use the temperature sensor and the internal voltage reference of 2.4V. There is a gain of 4 used in the programmable gain. If the value in the registers ADC0H:ADC0L is 0xAF00 and the data is left aligned, what is the corresponding temperature?

   Again, we need to obtain the input voltage to arrive at the temperature measured. The first step is to determine what the fraction of the full range voltage in the registers:

$$Ratio = \frac{AF0}{FFF} = \frac{2800}{4095} = 0.684$$

   *So, the voltage corresponding to this is 0.684*2.4V=1.64V. However, because we have a gain of 4, the voltage at the input is 1.64V/4=0.41V*

   The final step is to put this into the formula:

$$V_{TEMP} = 0.00286(TEMP_C) + 0.776$$

   Plugging in $V_{TEMP} = 0.41V$ *and rearranging gives the value of* $TEMP_C = -128°C$

3. If the measured room temperature is 10 degrees Celsius, the internal voltage reference of 2.4V is used, and the value in ADC0CF is 0x01, what is the voltage at the input of the ADC subsystem (i.e., before the MUX module) which would correspond to this temperature?

   The trick here was to realize that the gain really only matters for the number stored in the ADC registers. So the input voltage V_{TEMP} *can be obtained from the formula above as*

$$V_{TEMP} = 0.00286(10) + 0.776 = .8046$$

4. Assume that there is an array of 32 chars which is located at internal memory starting at location 0x32. Write a program which finds the largest and the second largest elements in this array. Put your results in the locations 80 and 81 in internal memory (you can assume that the chars represent the unsigned numbers from 0-255). There are a couple solutions for this program also

Below is a solution in C

```
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#pragma CODE
#include <c8051f020.h>                  // SFR declarations

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------

unsigned char idata array[32] _at_ 0x032;
```

```
unsigned char idata largest _at_ 0x80;
unsigned char idata larger _at_ 0x81;

//-------------------------------------------------------------------------------
// Function PROTOTYPES
//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------
// MAIN Routine
//-------------------------------------------------------------------------------
void main (void)
{
   // Declare all variables
        unsigned char i;
        unsigned char* pt;


   // Disable watchdog
   WDTCN = 0xDE;
   WDTCN = 0xAD;

        // Fill array
   for (i=0;i<32;i++)
        {
          array[i]=i;
        }

        pt = 0x32;      // Pointer to start of array
        largest = 0;    // All data are 0 or greater
        larger = 0;
        for (i=0;i<32;i++)
        {
          if (*(pt+i)>largest)
          {
            larger = largest;
            largest = *(pt+i);
          }
          else if (*(pt+i)>larger)
          {
            larger = *(pt+i);
          }
       }

  }
```

Below is a solution in assembler

```
;-------------------------------------------------------------------------
; Program Name: problem5asm.asm
; Author:Prof. hedrick
; Date: 03/28/2012
; Purpose: a soltuion to problem 5 in assignment 4
;-------------------------------------------------------------------------
#pragma debug list
$include (c8051F020.inc) ; Include register definition file

;-------------------------------------------------------------------------
; EQUATES
;-------------------------------------------------------------------------
GREEN_LED       equ P1.6
VALUES          equ 0x32
LARGEST         equ 0x80
LARGER          equ 0x81



;-------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-------------------------------------------------------------------------

; Reset Vector
   CSEG AT 00h
   ljmp Main     ; Jump beyond interrupt vector space.

;-------------------------------------------------------------------------
; CODE SEGMENT
;-------------------------------------------------------------------------
        cseg at 0x100
Main:   mov WDTCN, #0DEh   ; disable watchdog timer
        mov WDTCN, #0ADh
        mov SP, #2Fh       ; initialize stack pointer

// fill 32 bytes starting at 0x32 with test values
        clr A
                                        mov R0, #VALUES


fill:   mov @R0, A
                                        inc A
        inc R0
                                        cjne R0, #52h, fill
```

```
                                    nop


; R0 to get data value to compare,
; upper 128 byte of RAM, where LARGEST and Larger are tstored
; are accessable onlyby inderect addressing. Use R1 as a pointer store in
; tom RAM

        mov R0, #032h
        mov R1, #LARGEST
                                    mov @R1, #00
                                    mov R1, #LARGER
                                    mov @R1, #00

next:               mov R1, #LARGEST   ; move largest into TEMP to compare

                                    mov A, @R0
                                    xrl A, @R1                  ; check value
against largest
                                    jz  again       ; if are equal don't change
        mov a, @R0      ; reload value to check
        subb a, @R1     ;subtract from
                                    jc  less_than   ; value is less
                                    mov a, @R1      ;new value is gerater
; move largest value into larger
                                    mov R1, #LARGEST ;
                                    mov a, @R1      ; get largest value
                                    inc R1          ; R1 now points to  largest
                                    mov @R1, a      ; store largest in larger
; move new largest value from values to LARGEST
                                    mov a, @R0
                                    dec R1          ; R1 now points to largest
                                    mov @R1, a
                                    sjmp again
less_than:
; check to see if the current array value is greater than larger
        mov R1, #LARGER
                                    mov a, @R1
                                    xrl A, @R1
                                    jz again    ; values are equal
; value is greater than
        mov A, R0
        mov @R1,A


again:          inc R0
```

cjne R0, #0x52, next

nop

END

PROBLEM 5 Solution

```
/*
name: problem5.c
Author:Prof. Hedrick
Purpose: Read ADC0 and use collect 5 values, calculate the average and display the
average on the LCD display.

In order to have a reference voltage of 3 volts the external reference must be used.
NOTES: LED P1^6 causes UART 0 to stop working
*/

#pragma CODE

// Include  files
#include <c8051f020.h>
#include <stdio.h>

#define FALSE 0
#define TRUE 1


//-----------------------------------------------------------
// 16-bit SFE definitions for 'F02x
//-----------------------------------------------------------

sfr16 DP  = 0x82;     //data pionter
sfr16 TMR3RL  = 0x92;     //Timer3 reload value
sfr16 TMR3  = 0x94;     //Timer3 counter
sfr16 ADC0  = 0xbe;     //ADC0 data
sfr16 ADC0GT  = 0xc4;     //ADC0 greater than window
sfr16 ADC0LT  = 0xc6;     //ADC0 less than window
sfr16 RCAP2  = 0xca;     //Timer2 capture/reload
sfr16 T2  = 0xcc;     //Timer2
sfr16 RCAP4 = 0xe4;     //Timer4 capture/reload
sfr16 T4  = 0xf4;     //Timer4
sfr16 DAC0  = 0xd2;     //DAC0 data
sfr16 DAC1  = 0xd5;     //DAC1 data
```

```
//-----------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------

#define SYSCLK      22118400     //SYSCLK frequency in Hz
#define BAUDRATE    9600         //Baud rate of UART in bps
#define CMD 254
#define CLS  0x01



sbit LED = P1^6;                //LED

            //Pulse Width Modulation
//-----------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------

void SYSCLK_Init(void);
void config(void);
void UART0_Init(void);
void UART1_Init(void);
void ADC0Init(void);
char getUART0Char();
void delay(long i);
void sendUART1Char( unsigned char ch);
unsigned char num_to_ascii(unsigned char);
unsigned int getADC0Value(void);
void num_to_string(unsigned int number, char *num_str);
void average(unsigned int *values,unsigned int *avg,unsigned int *rem);
void display_str(char *in_str);


unsigned int xdata values[256] _at_ 0x00;

void main(void)
{

unsigned char outmsg[20];
int i;
unsigned int value[10];
unsigned int average_value;
char num_str[10];
```

```
unsigned int remainder;

        WDTCN = 0xde;    //disable watchdog timer
        WDTCN = 0xad;

        SYSCLK_Init();   //initialize oscillaor
        config();        // setup ports
        UART0_Init();    //initialize UART 0
        UART1_Init();    // initialize UART 1
        ADC0Init();      // initialize ADC0
 // clear LCD display
    sendUART1Char(CMD);
        sendUART1Char(CLS);



// get 10 values from the A/D port
   for (i = 0;i < 10; i++) {
     value[i] = getADC0Value();
   }

        average(value, &average_value, &remainder);
        num_to_string(average_value, num_str);
    display_str("Average: ");
        num_to_string(average_value, num_str);
        display_str(num_str);
        num_to_string(remainder, num_str);
        display_str("   Remainder:");
        display_str(num_str);



        for(;;);


}


void display_str(char *in_str) {
int i;
    i = 0;
        while (in_str[i] != NULL){
          sendUART1Char(in_str[i]);
          i++;
    }
}
```

```
// calculate the average 10 values
void average(unsigned int *values,unsigned int *avg,unsigned int *rem) {
int i;
unsigned int sum;
char num_str[10];
  sum = 0;
  for (i = 0; i < 10; i++) {
    sum = sum + values[i];
  }

  *avg = sum / 10;
  *rem = sum % 10;

}


// convert the A/D value to an ascii string
void num_to_string(unsigned int number, char *num_str) {
int i, j;
unsigned int remainder;
          j = 3;
          for (i = 0; i < 4; i++) {
            remainder = number % 10;
            number = number / 10;
      num_str[j] = num_to_ascii(remainder);
            j--;
   }
          num_str[4] = 0x00;
}


// convert a number to a single ascii char
unsigned char num_to_ascii(unsigned char num) {
unsigned char ascii_char;

   ascii_char = num + 0x30;
          return(ascii_char);
}


// send a single character to UART 1
void sendUART1Char( unsigned char ch) {
  SBUF1 = ch; // send character to UART transmit buffer
  while (!(SCON1 & 0x02)); // wait for transmit interupt flag to be set
```

```
                    // indicating that the character has been sent
  SCON1 &= 0xFC;   // interupt flag must be reset to 0 by software
}


// get a value from ADC0
unsigned int getADC0Value(void) {
unsigned int value;

    // start conversion
        AD0BUSY = 1;
        // wait for confversion to be complete
        while (AD0INT == 0);
        // read value
    value = ADC0;
        return(value);

}

/*
  wait here until a character is received from UART0
*/
char getUART0Char()
{
 char in_char;

  for (;;) {

    if (RI0 == 1) {
      in_char = SBUF0;
          RI0 = 0;
          return(in_char);
    }
  }
}

void delay(long i)
{
long j;
long delay_time = i*10000;
   for (j = 0; j < delay_time; j++);
}


//-------------------------------------------------------
// Config Routine
```

```
//Configured using Silicon Industries Configuration Wizard
//-------------------------------------------------------

void config(void){
//Configure the XBRn Registers

        XBR0 = 0x04; //XBAR0: Initial Reset Value
        XBR1 = 0x00; //XBAR1: Initial Reset Value
        XBR2 = 0x44; //XBAR2: Initial Reset Value

/*Select Pin I/O
Note: Some peripheral I/O pins can functionas either
inputs or outputs, depending on the configuration of the peripheral. By default,
the configuration utility will configure these I/O pins as push-pull putputs
*/
        // Port configuration (1 = Push Pull Output)
        P0MDOUT = 0x05; //output configuration for P0
        P1MDOUT = 0x40; //output configuration for P1
        P2MDOUT = 0x0F; //output configuration for P2
        P3MDOUT = 0x00; //output configuration for P3
        P74OUT = 0x00;  //output configuration for P4-7
        P1MDIN = 0xFF;  //Input configuration for P1
}


/*
initialization subroutines

SYSCLK_Init
This routine initializes the system clock to use an 22.1184 MHz crystal as its clock source
*/
void SYSCLK_Init(void)
{
        int i;                                  //delay counter
        OSCXCN = 0x67;              //start external oscillator with //22.1184 MHz crystal
        for (i=0; i<256; i++);      //wait for oscillator to start
        while(!(OSCXCN & 0x80));    //wait for crystal oscillator to settle was 0x88
        OSCICN = 0x08;              //select external oscillator as SYSCLK
                                                                //source and enable mission
clock detector was 88
}


/*
UART0_Init
Configure the UART0 using Timer1, for <baudrate> and 8-N-1
```

```
UART 0 uses P0 pin 0 for TX and P0 pin 1 for RCV
*/

void UART0_Init(void)
{
        SCON0 = 0x50;                           //SCON0: mode 1, 8-bit UART,
enable RX
        TMOD = 0x20;                            //TMOD: Timer1, mode 2, 8-bit
reload
        TH1 = -(SYSCLK/BAUDRATE/16);//set Time1 reload value for baudrate
        TR1 = 1;                                //start Timer1
        CKCON |= 0x10;                          //Timer1 uses SYSCLK as time base
        PCON |= 0x80;                           //SMOD00 = 1 (disable baudrate
divde-by-two
        TI0 = 1;                                //Indicate TX0 ready
}


/*
UART1_Init
Configure the UART1 using Timer 4, for <baudrate> and 8-N-1
UART 1 uses P0 pin 2 for TX ansd P0 pin 3 for RCV
*/

void UART1_Init(void)
{
        SCON1 = 0x50;                           //SCON1: mode 1, 8-bit UART,
enable RX
        SCON1 &= 0xFC;          // clear interupt pending flags
        T4CON = 0x30;
        RCAP4 = -(SYSCLK/BAUDRATE/32);
        T4 = RCAP4;
        CKCON |= 0x40;
        PCON |= 0x10;
        T4CON |= 0x04;
}

// initialize ADC0
void ADC0Init(void){
   //use DAC0 for Vref enable internal Vref and internal bias generator
   REF0CN = 0x13;
        // ADC configuration - enable ADC0, use AD0BUSY, right justified and AIN0
and AIN1
        // are single ended, PGA gain = 1
        AMX0CF = 0x60;
        AMX0SL = 0x00;   // select AIN0
```

```
        ADC0CF = 0xA8;    // ADC Configuration Register 22.118400MHz / MHz - 1 =
1, gain = 1
        ADC0CN = 0x80; // enable ADC0
        // configure DAC
        DAC0CN = 0x80;
        DAC0L = 0xFF;
        DAC0H = 0x0F;
}
```

Spring 2014