```
/*****************************************************************************
            Project: JHU ICX_Controller
-----------------------------------------------------------------------------
** File Name:       "ICX_main.c"
**
** Description:     Main control loop for ICX Electronics control software.
**
**  Author:         Joe Orndorff - Johns Hopkins University
**
** Creation Date:   Oct 19, 2004
** Revisions:
**
*******************************************************************************
**
** The following files are used and compiled with the Keil uVision2 Version 2.12 IDE:
**
** 89C51RD2.h
**
** These are UVision Files used by the compiler:
** STARTUP-RD2.A51
**
*******************************************************************************/

#include "ICX_Controller.h"
#include "globals.h"

/******************************/
    void main(void)
/******************************/
{
error_code=0;
while(1)
    {
    initialize_ICX();               // system initialization cmd
    event_handler();                // start user cmd handler
    }//end while
}
```

```
/*********************************************
 * File:  ICX_Controller.h
 *
 * Joe Orndorff / Johns Hopkins University.
  *********************************************/
#include "type_defs.h"
#include "89c51rd2.h"   // controller definitions
#include <stdio.h>      // prototype declarations for I/O functions
#include <intrins.h>    // instrinsic library
#include <stdlib.h>     // for type conversion functions 'atof()' etc
#include <string.h>

#define VERSION "VER 0.04" //software revision number

/* logic */
#define ENABLED 0
#define DISABLED 1
#define ON 1
#define OFF 0
#define TRUE 1
#define FALSE 0

/* PORT 0 assignments */
sbit  P07           =   P0^7;
sbit  P06           =   P0^6;
sbit  P05           =   P0^5;
sbit  P04           =   P0^4;
sbit  ICX2_DN_SNS   =   P0^3;
sbit  ICX2_UP_SNS   =   P0^2;
sbit  ICX1_DN_SNS   =   P0^1;
sbit  ICX1_UP_SNS   =   P0^0;

/* PORT 1 assignments */

sbit  SDA       =   P1^7;
sbit  SCL       =   P1^6;
sbit  P15       =   P1^5;
sbit  P14       =   P1^4;
sbit  P13       =   P1^3;
sbit  P12       =   P1^2;
sbit  P11       =   P1^1;
sbit  ICX2_ON   =   P1^0;

/* PORT 2 assignments */
sbit  P27   =   P2^7;
sbit  P26   =   P2^6;
sbit  P25   =   P2^5;
sbit  P24   =   P2^4;
sbit  P23   =   P2^3;
sbit  P22   =   P2^2;
sbit  P21   =   P2^1;
sbit  P20   =   P2^0;

/* port 3 assignments */
sbit P37        =   P3^7;
sbit P36        =   P3^6;
sbit P35        =   P3^5;
sbit P34        =   P3^4;
sbit P33        =   P3^3;
sbit ICX1_ON    =   P3^2;
sbit P31        =   P3^1;
sbit P30        =   P3^0;
/* hardware */

void travel_time(unsigned char);
void event_handler(void);
unsigned char decode_cmd(char*);

void initialize_serial(void);
void initialize_ICX(void);  // system initialization cmd
void initialize_89C51RD2(void);
//float Read_R2D(void);

/* utility function prototypes */
void delay (unsigned int);


/* I2C functions */
void i2c_start (void);                      //  Sends I2C Start Trasfer
```

```
void i2c_stop (void);                        //  Sends I2C Stop Trasfer
void i2c_write (unsigned char);              //  Writes data over the I2C bus
unsigned char i2c_read (void);               //  Reads data from the I2C bus
void i2c_rec_ack(void);
//void i2c_send_ack(void);                   // keep for later use
```

```
/*****************************************************************************
            Project: ICX_Controller
-----------------------------------------------------------------------------
** File Name:  "ICX_Commands.c"
**
** Description:  Table of ICX commands stored in code space
**               Note: This list must not exceed 25 items
**
**  Author:   Joe Orndorff - Johns Hopkins University
**
** Creation Date: OCT 22, 2004
** Revisions:
*****************************************************************************/
code unsigned char *ICX_Commands[25] =
{
"ICX1UP"           ,//    0
"ICX1DN"           ,//    1
"ICX2UP"           ,//    2
"ICX2DN"           ,//    3
"ICXSTATUS"        ,//    4
"ICXVERSION"       ,//    5
"ICXERROR"         ,//    6
"ICXRESET"         ,//    7
"               "  ,//    8
"               "  ,//    9
"               "  ,//   10
"               "  ,//   11
"               "  ,//   12
"               "  ,//   13
"               "  ,//   14
"               "  ,//   15
"               "  ,//   16
"               "  ,//   17
"               "  ,//   18
"               "  ,//   19
"               "  ,//   20
"               "  ,//   21
"               "  ,//   22
"               "  ,//   23
"               "   //   24
};
```

```
/******************************************************************************
            Project: ICX Controller
------------------------------------------------------------------------------
** File Name:       "I2C.c"
**
** Description:     functions to control the I2C bus
**                  These functions assume a uP clock of 11.0592MHz
**                  Delays may have to be scaled for faster clock
**
**  Author:         Joe Orndorff - Johns Hopkins University
**
** Creation Date:   Oct 19, 2004
** Revisions:
******************************************************************************/

#include "ICX_Controller.h"


/*************************************************************/
    void i2c_rec_ack(void)
/*************************************************************/
{
/* recieve acknowledge from slave */
SDA = 1;    //release the SDA line
SCL = 1;    // set clock high
_nop_();    // allow data to set up
SCL = 0;    // clock in ACK bit
}

///************************************************************/
//  void i2c_send_ack(void)
///************************************************************/
//{
///* send acknowledge to slave */
//SDA = 0;      // output ACK low level
//SCLK = 1;         // set clock high
//_nop_();      // allow data to set up
//SCLK = 0;     // clock out ACK bit
//}

/*************************************************************/
    void i2c_start (void)
/*************************************************************/
{
/* transmit start signal on I2C bus */
SDA = 1;        // Set data line high
while (!SDA);   // wait for SDA to go high
SCL = 1;        // Set clock line high
_nop_();        // delay
SDA = 0;        // Set data line low (START SIGNAL)
SCL = 0;        // Set clock line low
}

/*************************************************************/
    void i2c_stop (void)
/*************************************************************/
{
/* transmit stop signal on I2C bus */

SCL = 0;        // Set clock line low
SDA = 0;        // Set data line low
SCL = 1;        // Set clock line high
_nop_();        // delay
SDA = 1;        // Set data line high (STOP SIGNAL)
_nop_();        // delay
SDA = 1;        // Release SDA bus
SCL = 0;        // Nominal state
}

/*************************************************************/
    void i2c_write (unsigned char output_data)
/*************************************************************/
{
unsigned char index;

for(index = 0; index < 8; index++)          // Send 8 bits to the I2C Bus
    {
                                            // Output the data bit to the I2C Bus
    SDA = ((output_data & 0x80) ? 1 : 0);
```

```
    output_data  <<= 1;                     // Shift the byte by one bit
    SCL = 1;                            // Clock the data into the I2C Bus
    _nop_();
    SCL = 0;
    }
}

/*************************************************************/
    unsigned char i2c_read (void)
/*************************************************************/
{
unsigned char i,input_data;

SDA = 1;                // Release control of SDA line

input_data = 0x00;      // initialize
for(i=0;i<8;i++)        // Send 8 bits to the I2C Bus
    {
    input_data <<= 1;   // Shift the byte by one bit
    SCL = 1;            // Clock the data into the I2C Bus
    input_data |= SDA;  // Input the data from the I2C Bus
    _nop_();            // allow setup time
    SCL = 0;            // set clock low
    }
return input_data;
}
```

```
/******************************************************************************
            Project: JHU ICX_Controller
------------------------------------------------------------------------------
** File Name:       "Hardware.c"
**
** Description:     General hardware interface routines that
**                  do not fit nicely anywhere else
**
**  Author:         Joe Orndorff
**
** Creation Date:   October 22, 2004
** Revisions:
******************************************************************************/
#include "ICX_Controller.h"
#include "globals.h"

/****************************************/
  void initialize_ICX(void)
/****************************************/
{
/* set up Auxiliary Register */
AUXR        = 0x0D;     // AO=1 (disable ALE),XRS0=1,XRS1=1,M0=0
EA = 0;                 // Disable all interrupts
initialize_89C51RD2();  // initialize micro
initialize_serial();    // initialize serial port


//EA    = 1;            //enable selected interrupts
IT0 = 0;                //set int0 to level sensitivity
IT1 = 0;                // set int1 to level sensitivity
ICX1_ON = DISABLED;     //Make sure power up state has ICX1 relay turned off;
ICX2_ON = DISABLED;     //Make sure power up state has ICX1 relay turned off;
}

/*******************************************************************/
  void initialize_serial(void)
/*******************************************************************/
{
unsigned char i;

SCON    = 0x72; //8 bit UART

/*
 Set TI to enable TXSER UART function. Transmission Rate is 9600,N,8,1
 Formula:
 TH1=256-(K*OscFreq)/(384*BaudRate)
 K=0 if SMOD=0
 K=2  if SMOD=1
*/

TMOD    = 0x20; // Use TIMER1, mode 0, 8-bit auto-reload
TH1     = 0xFA; // Initialize Auto-reload to FA for 22.11MHz external clock
PCON    = 0x00; // when set we run at 19200 Baud, 9600 when clear
TCON    = 0X40; // Turn internal UART timer on

delay(20);      //allow 20ms for oscillator to start

/*Reset Serial Port*/
for(i=0;i<254;i++)
    {
    SCL = 0x01;         //SET SCL
    SCL = 0x00;         //CLR SCL
    }
SCL = 0x01;            //SET SCL
SCL = 0x00;            //CLR SCL
}

/*********************************/
  void initialize_89C51RD2(void)
/*********************************/
{
/*
this function cofigures the micro to
initial power on conditions
'0' sets i/o as ouput, '1' sets i/o as Input
*/

/*
initialize port 0 vairables
```

```
*/
    P0          =   0xFF;   //Use port 0 for data inputs

/*
 initialize port 1 variables
 */
    SDA         =   0; // I2C Serial Data/Address (output)
    SCL         =   0; // I2C Serial Clock (output)
    ICX2_ON     =   0; // ICX2 Control (output)

/*
 initialize port 2 variables
*/
    P2          =   0x00; //LCD output port (outputs)

/*
 initialize port 3 variables
*/
    P3          =   0xFF;   // Use the alternative function of Port 3
    ICX1_ON     =   0;      // ICX1 Control (output)

/*
  set up Auxiliary Register
*/
    AUXR        =   0x0D;   //AO=1 (disable ALE),XRS0=1,XRS1=1,M0=0
}
```

```
/*****************************************************************************
            Project: JHU ICX_Controller
-----------------------------------------------------------------------------
** File Name:       "Global_vars.c"
**
** Description:     Keeping these nasty things in their own file
**
**
**  Author:         Joe Orndorff
**
** Creation Date:   October 22, 2004
** Revisions:
*****************************************************************************/
#include "type_defs.h"

/* STATUS REGISTER IN BIT SPACE */
unsigned char bdata     status_reg;     // general 8 bit status register


/* ERROR REGISTER IN BIT SPACE */
unsigned char bdata     error_reg;      // gerneral 8 bit error register
sbit error          =   error_reg^0;

/* GLOBALS IN DATA RAM */

/* GLOBALS IN XRAM */
xdata unsigned char error_code;
```

```
/*****************************************************************************
            Project: JHU ICX Controller
-----------------------------------------------------------------------------
** File Name:       "EventHandler.c"
**
** Description:     Handles commands received via the serial port
**
**  Author:         Joe Orndorff - Johns Hopkins University
**
** Creation Date:   Oct 19, 2004
** Revisions:
**
******************************************************************************/
#include "ICX_Controller.h"
#include "globals.h"
#define TIMEOUT 1000
/*******************************/
 void event_handler(void)
/*******************************/
{
xdata char InString[50] = ""; //string input from serial port
xdata unsigned char reset = 0; //loop terminator
xdata unsigned char cmd = 0;
unsigned char status = 0;

while(!reset) //exit and reset on a ^C
    {
    gets(InString,20);
    cmd = decode_cmd(InString);
    switch(cmd)
        {
        case 0: //ICX1UP
            if(ICX1_UP_SNS == DISABLED)
                ICX1_ON = ENABLED;
            travel_time(1);
            printf("\n");
            break;
        case 1: //ICX1DN
            if(ICX1_DN_SNS == DISABLED)
                ICX1_ON = DISABLED;
            travel_time(2);
            printf("\n");
            break;
        case 2: //ICX2UP
            if(ICX2_UP_SNS == DISABLED)
                ICX2_ON = ENABLED;
            travel_time(3);
            printf("\n");
            break;
        case 3: //ICX2DN
            if(ICX2_DN_SNS == DISABLED)
                ICX2_ON = DISABLED;
            travel_time(4);
            printf("\n");
            break;
        case 4: //ICXSTATUS
            status = ~P0 & 0x0F; // sensor bits
            status = (error_code * 16) | status; // error code (1 to f)
            printf("%2.2bx\n",status);
            error_code = 0; //clear error since it was read
            break;
        case 5: //ICXVERSION
            printf("%s\n",VERSION);
            break;
        case 6: //ICXERROR
            printf("%bx\n",error_code);
            error_code = 0;
            break;
        case 7: //ICXRESET
            reset=1;
            break;
        default:
            printf("?\n");
            break;
        } // end switch
    } // end while
}

/****************************************/
```

```
 unsigned char decode_cmd(char string[50])
/****************************************/
{
unsigned char i;
for (i=0;i<25;i++)
    {
    if (strcmp(string,ICX_Commands[i])==0)
        return(i);
    }
}

/****************************************/
  void travel_time(unsigned char select)
/****************************************/
{
xdata unsigned int timeleft = TIMEOUT;
bit sensor=DISABLED;

while(sensor == DISABLED && timeleft)
    {
    delay(10); //wait 10ms
    /*
      issue busy char if MCC trying to send cmd
    */
    if(RI)
        {
        _getkey();
        putchar('~');
        }
    timeleft --;
    switch(select)
        {
        case 1:
        sensor = ICX1_UP_SNS;
        break;
        case 2:
        sensor = ICX1_DN_SNS;
        break;
        case 3:
        sensor = ICX2_UP_SNS;
        break;
        case 4:
        sensor = ICX2_DN_SNS;
        break;
        }//end switch
    }//end while

if(sensor != ENABLED)
    error_code = 0x01;

}
```

```
/*******************************
 * Utilities.c                  *
 * Rev 6-18-2004                *
 * Johns Hopkins University     *
 *******************************/
#include "ICX_Controller.h"
#include "globals.h"

/*****************************************/
  void delay (unsigned int milliseconds)
/*****************************************/
/*
  this function assumes an oscillator
  frequency of 22.118Mhz
  to calculate TH0 and TL0 register
  values:
  assume a delay of 1ms
  TH0 is the High Byte of the Timer register
  HL0 is the Low Byte of the Timer register
  TH0,TL0 = 65536 - ((delay x OSC frequency)/12)
  convert to hex and divide the bytes between
  the TH0 and TL0 registers

 THIS DELAY FUNCTION CAN NOT RUN IN AN 'ISR' -Interrupt Service Routine
*/

{
unsigned int i;

TMOD |= 0x01;                    // configure timer 0 for 16 bit
    for(i=0;i<milliseconds;i++) // wait #of milliseconds
        {
            TH0 = 0xF8;
            TL0 = 0xCD;          // load the 16bit timer low Byte
            TR0=1;               // start timing
            while (!TF0);        // Wait for Overflow to be set
            TF0=0;               // reset overflow
            TR0=0;               // stop timer
        }
}
```