A PIC-based LCD Display for Stand-Alone Instrumentation

Michael Case, Dr. Bruce E. Segee
Department of Electrical and Computer Engineering
University Of Maine
Instrumentation Research Laboratory

Abstract

A recent application in instrumentation led to the development of a temperature controller unit that was capable of operating independently or in a shared RS-232 arrangement under control of a host computer. However, when this system is used independently, no user interface is available other than a set of DIP switches to control the set point. Using a PIC 16C773 microcontroller and a conventional two-line LCD display we have developed a display device whose functionality is very flexible. The device uses the Serial Peripheral Interface (SPI) to interrogate one or more controllers and cycle the contents of the LCD display. The display unit is configured as an SPI master (one who can initiate the transfer of information).
Multiple slave devices can be connected to a single SPI. Thus, one can easily have one device per display, one display for many devices, or a mixture of the two. While this device was developed for a specific application, its usefulness extends to a broad range of applications.

1. Introduction

1.1 Problem Description

In a joint effort between the University of Maine's Instrumentation and Research Laboratory and Sensor Research and Development Corporation, a device used to control the temperature of solid state thin film gas sensors has been in development. A block diagram of the sensor temperature control circuitry is shown in Figure 1. A host computer can interact with the system using a shared RS-232 network to set or check temperature and many other things, but a goal of the system is to also allow independent operation. When running independently, the only form of interaction between the user and the temperature controller is by means of a set of DIP switches that can be used to specify a temperature setting. Although the system is precalibrated to provide precise temperature control, confirmation of temperature and continual monitoring is highly desirable. A method of data collection and presentation needed to monitor the status of the sensors was required.

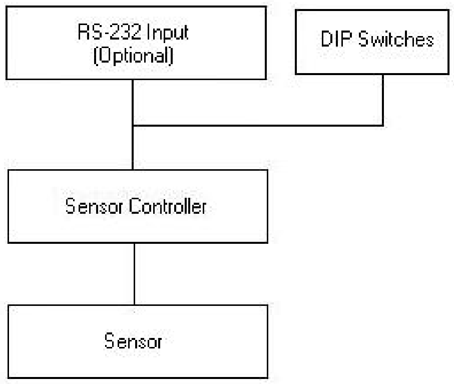
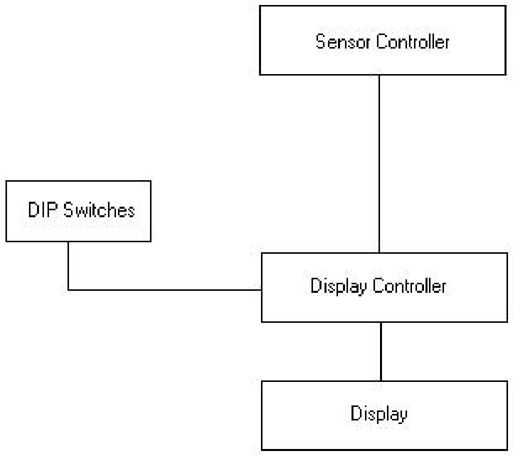


Figure 1. Block Diagram of sensor controller circuit

1.2 The Solution

It was decided that a microcontroller would be employed to interface to the temperature controller and an LCD display unit would be used to display information collected from the sensors. In addition, the unit was to be completely stand-alone and was also to be easy to configure for other applications. The developed stand-alone unit is controlled by a PIC16C773 microcontroller with Serial Peripheral Interface (SPI) support. In this particular application the unit displays the identification number of the sensor it is monitoring and the temperature of that particular sensor. A block diagram of the display system is shown in Figure 2.

Figure 2. Block Diagram of display circuit



1.3 Concerns

It was necessary to develop a system that was low-cost, low-power and small in size. It was also critically important, however, that the system not interfere with the temperature controller.

Finally the display unit had to be very flexible. The design was so that a display unit could interface to either a single temperature controller or that one display could be used for many controllers. Finally, it was necessary that the presence or absence of a display device not affect the controller, i.e., a controller may also see zero display devices.

2. Implementation

This section focuses on the concerns in the previous sections and how they are being addressed.

2.1 Circuit Design

The circuit is comprised of several readily available pieces of equipment. A schematic diagram of the system is shown in Figure 3. The main components of the circuit are the PIC, LCD display, and DIP switches.

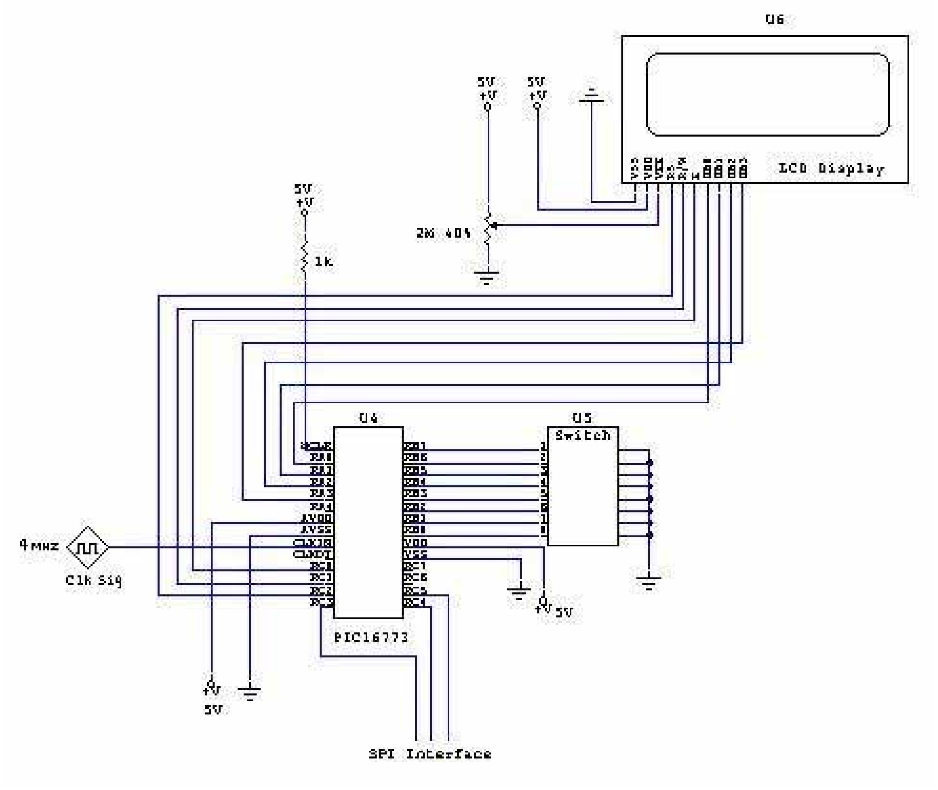


Figure 3. A schematic diagram of the display

The LCD display unit is a backlit 2 row by 20-character dot matrix display. The connections needed for the display include +5 volts, ground, contrast adjustment, a 3-bit control bus and the data bus connections. In its current configuration a 4-bit data transfer is used so only the four least significant bits in the data bus are required. For contrast adjustment a potentiometer is connected to +5 volts and set at approximately 40%. The data transfer lines and control lines are connected to the PIC controller.

The PIC is a PIC16C773 in a DIP package. The pinout for this package is shown in Figure 4. The connections (See Figure 4) include the basic +5 volts and ground connections. Also, a 4

Mhz crystal was used as a clock input and is included in the circuit to further support the independent theme. Port B (RB0-RB7) was used as the input for the DIP switch, since it has internal pull-up resistors. Three Port C bits (RC0-RC2) were used for the displays control bits: Register Select, Read/Write, and Enable. Four bits on Port A (RA0-RA3) were used for the display's data bits. Finally, three bits on Port C (RC3-RC5) were used for SPI communication. The PIC for the display circuit is set as the SPI master

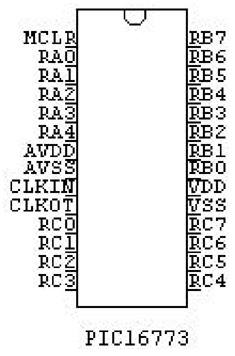


Figure 4. Pinout of a PIC16C773

Finally the 8-switch DIP switch is simply connected to the PIC on one side, and to ground on the other side. The eight DIP switches are used in two groups of four as shown in Figure 5. These specify two four-bit numbers that represent the ID number of a controller that is the start of a sequence and the end of the sequence, respectively. The display continuously cycles at a 1 Hz refresh rate through all the controllers from the beginning to the ending ID. The switches are used in the software that controls the PIC microcontroller. No external pull-up resistors are required since the PIC microcontroller is capable of supplying weak pull-up resistors on Port B, intended for this exact purpose.

2.2 PIC Software

The software on the PIC for this application was written in the C language and was compiled and programmed into the PIC. To make the software easy to apply to many different applications, the common pieces of code that would always be needed are written in functions to clean up the main body of code. For the current application the main body of code checks the DIP switches to find the number of sensors, and which sensors to check. The four least significant bits on the DIP switch determine a starting sensor number (1-15) while the other four bits determine an ending sensor number (1-15). The number desired is selected by orienting the switches to create a binary value.

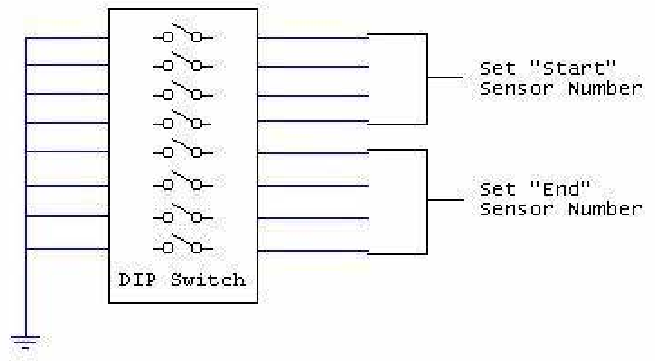


Figure 5. DIP Switch operation

The PIC uses the Serial Peripheral Interface (SPI) port to call the respective sensor and “ask” for the temperature. The display is set as the SPI master and is very flexible. Not only can a single slave be connected to the SPI port, but many other slaves can also be connected, however only one can be accessed at one time. The combination of slaves and masters can vary however the developer wishes. Two possible configurations are shown in Figure 6.

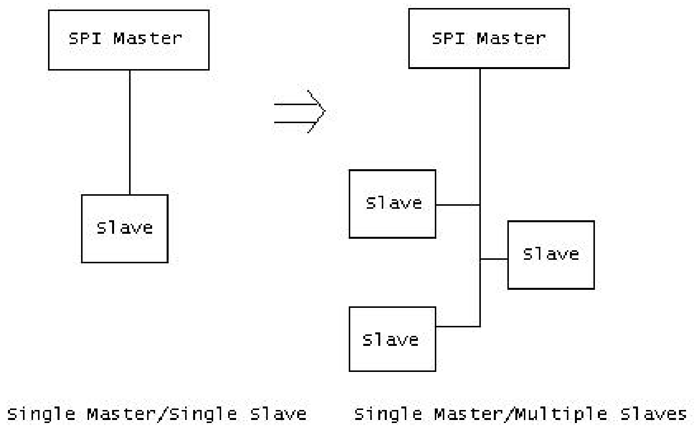
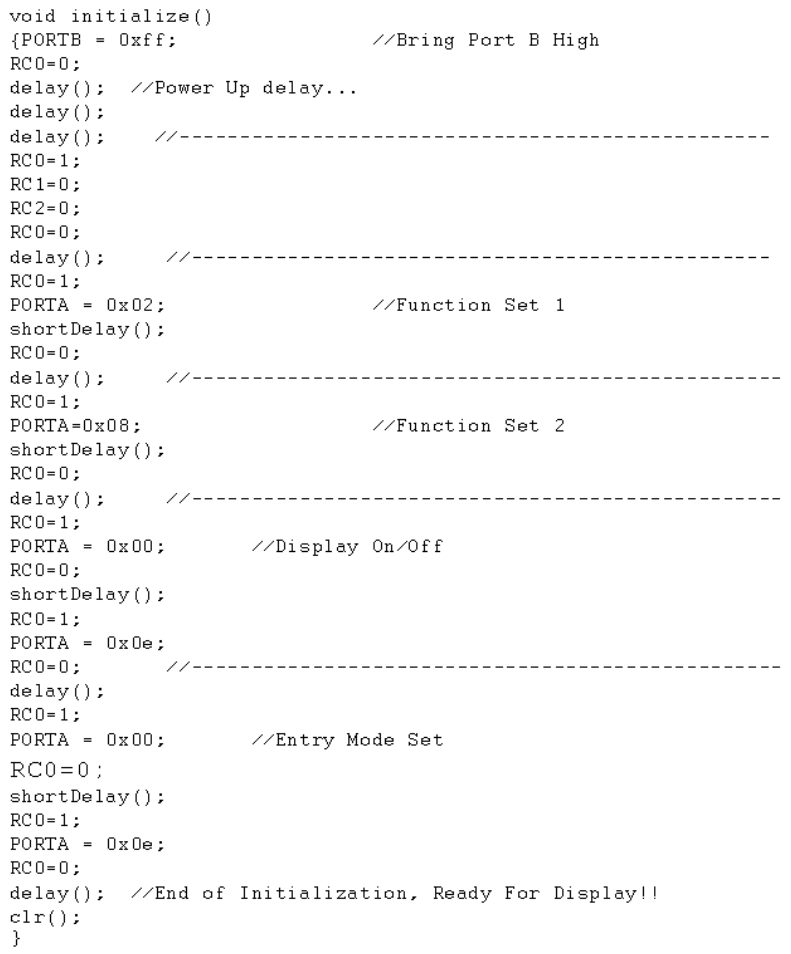


Figure 6. Possible SPI Configurations

When the software begins operation, the display is initialized. The code for the initialization is shown in Figure 7. In its current configuration, the software configures the display for two-line display and 4-bit data transfer. In the initialization code in Figure 7, the Enable bit for the display is designated by RC0, and is manipulated by manually setting it at either 0 or 1. The Register Select bit is represented by RC1 and the Read/Write bit is represented by RC2. Delays in the form of count-down functions are used between instruction blocks to allow the display time to set up. The “Function Set 1” block sets the display for 4-bit data transfer. The “Function

Set 2" is used to set the display for two line output. The final two blocks turn on the display and set it to accept instructions for data output.
After initialization, the software begins running in the main loop, as is shown in code in Figure 8. While the software is running in this loop, the sensor number and temperature of the current sensor is displayed for about a second. Depending on the DIP switch settings sensors are cycled through in a loop, each being displayed for a second. In the beginning of the loop, the DIP switches are checked and the starting ID ("lowSensorNum" in the code) and the stopping ID ("highSensorNum" in the code) are stored. If no DIP switches are selected for a beginning point then the display is blank and the software continues to loop until a beginning point is selected. The temperature readings are taken just before display by the "getTemperature" function that calls the sensor with current ID and "asks" for the temperature. An sprintf command is used to put the output information into a string, and then the "outToDisplay" function sends all the information in the string to the display. The "outToDisplay" function is shown in Figure 9. The code runs through the string putting the information on the data bus in two pieces. Since a 4-bit data transfer is being used, each character has to be sent in two pieces. When all the information of the string has been output, the rest of the display receives blank characters.



```
void initialize()
{PORTB = 0xff;                //Bring Port B High
RC0=0;
delay();  //Power Up delay...
delay();
delay();    //--------------------------------------------------
RC0=1;
RC1=0;
RC2=0;
RC0=0;
delay();     //-------------------------------------------------
RC0=1;
PORTA = 0x02;                 //Function Set 1
shortDelay();
RC0=0;
delay();     //--------------------------------------------------
RC0=1;
PORTA=0x08;                   //Function Set 2
shortDelay();
RC0=0;
delay();     //--------------------------------------------------
RC0=1;
PORTA = 0x00;       //Display On/Off
RC0=0;
shortDelay();
RC0=1;
PORTA = 0x0e;
RC0=0;       //--------------------------------------------------
delay();
RC0=1;
PORTA = 0x00;       //Entry Mode Set
RC0=0;
shortDelay();
RC0=1;
PORTA = 0x0e;
RC0=0;
delay();  //End of Initialization, Ready For Display!!
clr();
}
```

Figure 7. Initialization code

```
while(1)            //Loop outputs information, refreshing each
                                        //time through the loop.
          {
                    clr();
                    highSensorNum = (PORTB & 0xF0)>>4;
                    lowSensorNum = PORTB & 0x0F;

                    if(index > highSensorNum) index = lowSensorNum;   //After checking last, start again
                    temp = getTemperature(index);           //Check sensor temperature

                    if(highSensorNum && lowSensorNum != 0)  //If DIP is set at zero, wait until changed.
                    {
                     sprintf(buffer, "%s %d", firstString, index);
                     outToDisplay(buffer);
                     nextLine();
                     decimalTemp = temp;
                     decimalTemp = decimalTemp/10;
                     sprintf(buffer, "%s %d.%d", secondString, decimalTemp, floatingTemp);
                     outToDisplay(buffer);
                  }
                   waitOneSecond();     //Display data for about a second

                     index++; //Ready for next sensor
          }
```

Figure 8. Main code loop

```
void outToDisplay(char *buf)
{
                    int count = 0;
          while(count < LCDwidth)
          {
                    if(count < strlen(buf))
                    {
                      print(buf[count]>>4);
                      print(buf[count] & 0x0f);
                    }
            else
                    {
                      print(2);
                      print(0);
                    }
            count++;
          }
}
```

Figure 9. Data output function

2.3 Education

Above all, while constructing this device we were able to see how an LCD Display operates and what is required of it. Also, some of the features of the C coding language were learned by a rather inexperienced programmer. From an educator's perspective the project would be an excellent way to demonstrate the tools required to program an LCD Display and even the circuitry required to operate a core display.

3. Conclusion

We have developed a piece of equipment that will provide much needed interaction to the sensor project. The circuit design also makes for a very flexible device that can be used for many other projects. Not only is the software easy to change but the circuitry itself can be modified to adjust to a variety of uses.

Bibliography
1. Savitch, W *Problem solving with C++*, Addison Wesley Publishing, (1999)
2. URL: http://www.microchip.com/10/lit/pline/picmicro/families/16c77x/index.htm; PIC16C77X Family
3. Schildt, H *C++, The Complete Reference*, McGraw Hill (1991)

MICHAEL CASE
Michael J. Case is currently an undergraduate student at the University of Maine pursuing a degree in Electrical Engineering. He currently works in the Instrumentation Research Laboratory at the University of Maine part time during the semester.

BRUCE SEGEE
Bruce E. Segee is an Associate Professor of Electrical and Computer at the University of Maine. His research interests include Instrumentation, Automation, and Intelligent Systems. He is the Director of the Instrumentation Research Laboratory and a Member of the Intelligent Group at the University of Maine. His work focuses on real-world deployable systems for use in manufacturing environments. Dr. Segee received his PhD from the Department of Electrical and Computer Engineering at the University of New Hampshire in 1992.