Microprocessor Networking with a Minimum Number of External Connections

Dr Bruce E. Segee (email: segee@eece.maine.edu),
Binaya Acharya (email: bacharya@eece.maine.edu),
Isaac Horn (email: isaac.horn@umit.maine.edu),
Michael Case (email: michael.case@umit.maine.edu)

Department Of Electrical and Computer Engineering
Instrumentation Research Laboratory
University of Maine, Orono

Abstract

Networking refers to the connection of multiple systems together to allow the transfer of information to and from each other. This can be achieved in a variety of ways. Such methods include serial connections such as RS-232 or RS-485 as well as more sophisticated methods such as IEEE 802.3 (commonly called Ethernet). Furthermore networking between the systems can use a wire connection, fiber optics, or wireless connections.

A networking of microprocessors using Ethernet is being developed at the Instrumentation Research Lab, University of Maine. Owing to the design constraints, it is necessary to have a network of a large number of microprocessors that are located inside an enclosed space. The application requires that power be supplied externally. Overall, the number of connections to the outside world must be a minimum. Each microprocessor interacts with a variety of different I/O devices, both analog and digital. After a thorough research of all the available microprocessors, the RabbitCore module RCM2200 was selected. This is a very small board that includes the Microprocessor as well as a built-in Ethernet compatibility. Furthermore, the inclusion of a TCP/IP protocol stack allows the network to easily interface to the outside world. Including power and ground, the number of connections to the outside world is six, yet the number of controllers, processing power, or number of I/O points is virtually unlimited. Furthermore, the use of standard networking protocols simplifies the software design and improves the flexibility.

1.Introduction

Microprocessors have long been used to control devices but their power to transfer data over a network has seldom been exploited. The use of microprocessor

networking is highly desired when data from small components like a temperature sensor needs to be sent over to a different location. Typically, when information from a component like a sensor needs to be transferred, a PC collects the information from the sensor via a microprocessor and relays it to a server. As illustrated in Figure 1A, the server then sends the information using Ethernet or any other preferred networking mechanism to the client. But, it would be cheaper and efficient if we could use a microprocessor or a group of microprocessors to do all the information gathering and relaying to the clients. Such a networking was viable using a Rabbit 2200 microprocessor module. Figure 1A and Figure 1B illustrate the comparison between the typical networking and the preferred networking using the Rabbit 2200 module.

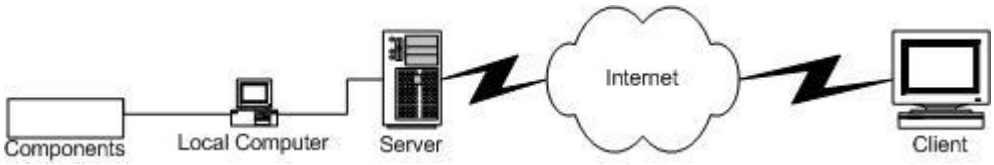

Figure 1A. A typical networking

Figure1B. The preferred method using the Rabbit2200

The Rabbit 2200 module was the microprocessor of our choice after a thorough research on the available microprocessors. This is a very small board that includes the Microprocessor as well as a built-in Ethernet compatibility. Furthermore, the inclusion of a TCP/IP protocol stack allows the network to easily interface to the outside world. Including power and ground, the number of connections to the outside world is six, yet the number of controllers, processing power, or number of I/O points is virtually unlimited. Furthermore, the use of standard networking protocols simplifies the software design and improves the flexibility. A picture of a Rabbit 2200 is shown in Figure 2.
.

Figure 2. A Rabbit 2200 module beside a penny.

2.The Need

The fundamental need for the project is to be able to make a number of microprocessors talk to each other and to the world using minimum possible number of connections. This networking has to be cheap and affordable and at the same time very versatile to be implemented on any system, whether it be measuring temperature using temperature sensors or monitoring a home. The external circuitry should be small owing to the design constraint that the system needs to be in an enclosed space as illustrated by Figure 3.

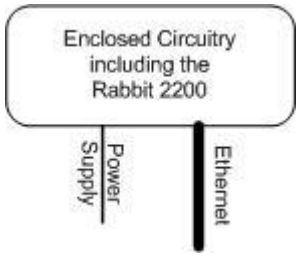

Figure 3. An Ideal System

3.An Application: Lab Temperature on the Web

A demonstration of the networking was made using a Rabbit Module, a temperature sensor, ADC and a webpage. This application measured the laboratory temperature and displayed it on a webpage served by the Rabbit 2200 module. Refreshing the webpage displays the temperature at that particular moment. The application setup on a prototyping board is illustrated on Figure 4. A sample web page is shown in Figure 7 and discussed in section 3.3.

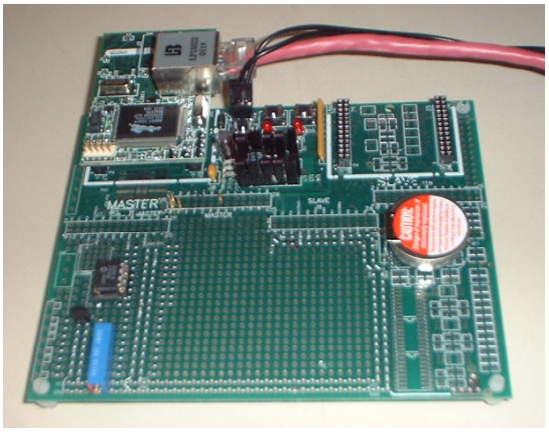

Figure 4. The Application on a prototyping board.

This application has two modules.

1. Temperature Measurement
2. Web Output

3.1 Temperature Measurement

The temperature measurement circuitry uses LM35DZ temperature sensor and MCP3202 Analog to Digital Converter. Our sensor circuitry was very basic and used a 5V supply. Figure 5 shows this circuitry. The sensor is linearly calibrated to a 10mV/ °C scale factor and is small, cheap and simple to use.

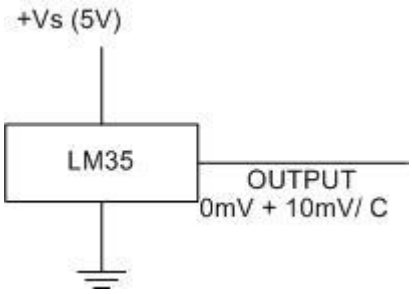

Figure 5. The basic sensor circuitry

The MCP3202, the ADC of our choice, comes in an 8-pin dip package. Its relatively small size makes it a good fit for our project. The SPI (Serial Peripheral Interface) compatibility made it easier and lowered the number of connections to the Rabbit Module. A pin diagram for the MCP3202 is shown in Figure 6.

The analog output from the sensor was fed to the ADC. The ADC converts the analog signal from the sensor to a 12 bit digital number. This digital value is a number between 0 and 4095 corresponding to the 0 and 5 V supply so each ADC count corresponds to 0.00122 V changes in voltage. Thus, approximately 8 ADC counts represent 1ºC change in temperature.

As the sensor is linearly calibrated to 10mv/ºC, the temperature in ºC can be calculated using the following relationships.

Temperature in ºC = Output Voltage from the sensor * 100

OR

Temperature in ºC = ADC value * 0.122

The basic relationship between the Celsius and Fahrenheit is used to convert the temperature into the Fahrenheit scale.

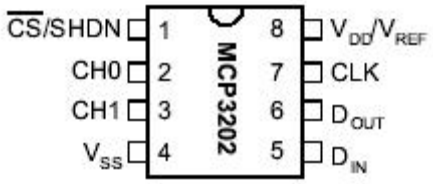

Figure 6. The pin diagram for the MCP3202

3.2 Web Output

The temperature is then passed to a website that is served by the Rabbit server. The server program is essentially a C code application that has all the information about the files and pictures that are to be hosted by the little server. A sample of the output of this server is shown in Figure 7.

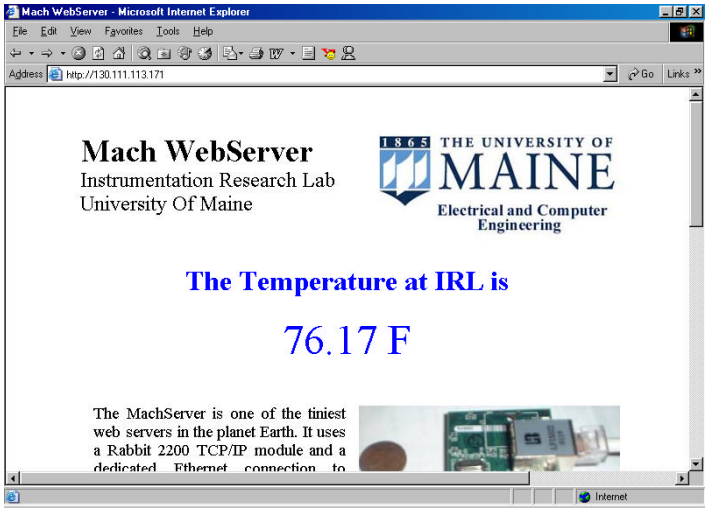

Figure 7. The Website

4.The C Code

The following are the highlights of the C code for the application. This includes the server code and the ADC conversion code. The codes are commented as they occur.

```
//Setting up the IP address, netmask and gateway
#define MY_GATEWAY     "130.111.113.10"
#define MY_IP_ADDRESS  "130.111.113.171"
#define MY_NETMASK     "255.255.0.0"

//defining the library and memory map
#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"

//Importing the files to the server
#ximport "C:\WINDOWS\Desktop\Website\INDEX_FILE.sHTML"    index_html

//Data structure HttpType from the http.lib associates any file
extension with a MIME type.
const HttpType http_types[] =
{
   { ".shtml", "text/html", shtml_handler},
};
```

```
//the variables
int temp_val;
char temp[20];

//Data structure HttpSpec contains all the files, variables and
functions the web server has access to.

const HttpSpec http_flashspec[] =
{
   { HTTPSPEC_FILE,"/",index_html,NULL, 0, NULL, NULL},
   { HTTPSPEC_FILE,"/index.shtml",index_html, NULL,0, NULL, NULL},
};

//Main routine initializes the socket and server. It also calls the ADC
routine that does the conversion.
main()
{
  double voltage;
   sock_init();
   http_init();
      init();
   tcp_reserveport(80);

  while (1) {
      temp_val = read();
      voltage=temp_val*5.0/4096.0*100;

      sprintf(temp,"%6.2lf",(1.8*voltage)+32);
      http_handler();
   }
}

//This program clocks and RW's so that the Rabbit can use a MCP3202 ADC
/*
PIN ASSIGNMENTS
PA0 - chip select
PA1 - clock
PA2 - data to ADC
PB5 - input
*/
init (){ //makes all portA pins low, taking care of the !CS, initiating
communication with ADC
      WrPortI(SPCR, &SPCRShadow, 0x84); //initialize A as output port
      WrPortI(PADR, &PADRShadow, 0x01); //make portA bits low
}//init


clock (){
      BitWrPortI(PADR, &PADRShadow, 1, 1);  // turn PA1 on (clock high)
      BitWrPortI(PADR, &PADRShadow, 0, 1);  // turn PA1 off (clock low)
}//clock


send (int val){ //sends bits to ADC's Din pin
      if (val)
            {BitWrPortI(PADR, &PADRShadow, 1, 2);}
```

```
      else
            {BitWrPortI(PADR, &PADRShadow, 0, 2);}
      clock ();
}//send


int accept (){ //simple routine used by a more complicated 'read'
routine
      clock ();
      return BitRdPortI(PBDR, 5);
}//accept


int read (){ //sends start bit to ADC, and configures for 'single-
ended' reading of channel 1
            //uses accept() to read a 12-bit stream of data
            //data comes from the device in MSB-first format
      int i;
      int val; val = 0;

      BitWrPortI(PADR, &PADRShadow, 0, 0); //!CS low, chip selected

      send(1); //start bit to MCP3202
      send(1); send(1); //configure MCP3202 for single-ended mode
reading, from channel 1
      send(0); //make the MSBF feature of the MCP3202 low

      for(i=0; i<12; i++){ //read a 12-bit stream of data, minus NULL
bit
                           //first accepted bit is a NULL bit
            val = accept() | (val<<1);
      }//for
      BitWrPortI(PADR, &PADRShadow, 1, 0); //!CS high, chip deselected
      return val;
}//read
#nodebug
```

5.Conclusion

The Rabbit 2200 module is a highly useful component. Its small size, low price, ease of programming and networking, makes it a good choice for a variety of uses. The board communicates using IEEE 802.3 (10 Base T Ethernet) and includes TCP/IP. This makes it relatively easy to perform tasks normally reserved for a dedicated server, or at least a desktop PC, such as serving web pages, and sending and receiving E-mail. Our goal in this is to create a network of controllers with a minimum number of connections between it and the outside world, but the board is equally well suited to a wide variety of goals. In particular, the fact that it is programmed in C and provides access to a wide range of protocols and is easily interfaced to custom hardware make it an ideal teaching tool as well as research tool.

6.Bibliography:

1. Dynamic C TCP/IP User's Manual, 2001 Z – World Inc, Davis, CA
http://www.rabbitmicro.com/documentation/docs/TCPIP_ref/User/index.htm

2.User's Manual for Integrated C Development System, 2001 Z – World Inc, Davis, CA
http://www.rabbitmicro.com/documentation/docs/dc_premier/dcm1.htm

3. Data Sheet for MCP3202 ADC. 1999 Microchip Technology Inc,
http://www.microchip.com/1010/pline/analog/anicateg/mixed/signal/adc/sar/devices/mcp3202/7027/index.htm

4. LM35 Precision Centigrade Temperature Sensors. 2000 National Semiconductor Corp.
http://www.national.com/pf/LM/LM35.html#Datasheet

5.Bill Giovino, "Zilog and the Embedded Internet- a White Paper" http://Microcontroller.com 09/22/200

6. Mark Thoren "TCP/IP Using Microcontrollers and the Development of and Internet mp3 Player" An Independent Study Report, ECE 599 Spring 2001, University Of Maine.

.

BRUCE SEGEE
Bruce E. Segee is an Associate Professor of Electrical and Computer Engineering 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 Systems 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 University of New Hampshire in 1992.

BINAYA ACHARYA
Binaya Acharya is currently enrolled as an undergraduate student at the University of Maine as an Electrical Engineering major. He has been working in the Instrumentation Research Lab, Department of Electrical and Computer Engineering at the University of Maine as an Undergraduate Research Assistant since Feb 2000.

ISAAC HORN
Isaac Horn is currently enrolled as an undergraduate student at the University of Maine as an Electrical Engineering major. He has been working in the Instrumentation Research Lab, Department of Electrical and Computer Engineering at the University of Maine as an Undergraduate Research Assistant since June 2000.

MICHAEL CASE
Michael Case is currently enrolled as an undergraduate student at the University of Maine as an Electrical Engineering major. He has been working in the Instrumentation Research Lab, Department of Electrical and Computer Engineering at the University of Maine as an Undergraduate Research Assistant since Oct 2001.