Engineering Bulletin

EB390/D
5/2002

Porting the AN2120/D UDP/IP Code to the Avnet Evaluation Board

By: Steven Torres
Motorola
SPS TSPG 8-/16-Bit Division

Introduction

The Motorola application note *Connecting an M68HC08 Family Microcontroller to an Internet Service Provider (ISP) Using the Point-to-Point Protocol* (AN2120/D) describes a methodology for connecting a Motorola microcontroller to the Internet using the PPP protocol to exchange UDP/IP data. UDP/IP (user datagram protocol/Internet protocol) is similar to TCP/IP, except that being a connectionless protocol, it sends messages to a host without establishing a connection.

AN2120/D also includes the code that implements a UDP/IP protocol. This code was first developed using the Cosmic development environment. Since then, the UDP/IP code has been ported to Metrowerks® CodeWarrior® development environment[1] and onto the 68HC908GP32 Avnet evaluation board.

The UDP/IP methodology provides a mechanism to allow a remote device, in this case an M68HC08 Family microcontroller, to connect to an ISP. When the remote device has established a connection with the ISP, the remote device can broadcast data to other hosts on the Internet. The UDP/IP network connectivity is illustrated in **Figure 1**.

1. CodeWarrior is a registered trademark of Metrowerks, a Motorola company.

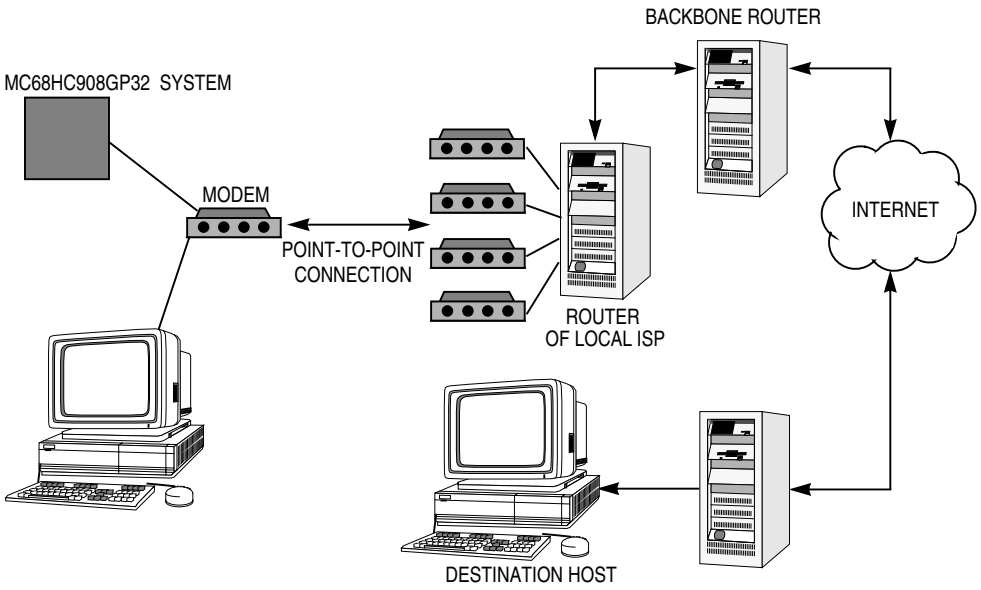


Figure 1. Network Framework

Porting Overview

This engineering bulletin describes the porting of the UDP/IP CodeWarrior code to the Avnet evaluation board. The information provided in subsequent sections is organized as a guide to connecting Motorola microcontrollers to the Internet using the UDP/IP code.

The demo's objectives are:

- Connecting an Avnet evaluation board running a CodeWarrior version of the UDP/IP code on its MC68HC908GP32 MCU to a dial-up server through a PPP link
- Demonstrating how an ActiveX® component can be used as a destination host to interface with the device running the UDP/IP code through a dial-up server

NOTE: *This engineering bulletin does not detail the SLIP connection already described in AN2120/D.*

In subsequent sections, this engineering bulletin describes:

- The Avnet evaluation board hardware
- UDP/IP CodeWarrior code modifications that are required for the demo
- How to organize, set up, and configure a network to be compatible with a PPP connection
- Setting up an ActiveX interface on the destination host to interface with the remote device
- Some techniques and tools for debugging an Internet connection between an MCU and a destination host

While this engineering bulletin details using a dial-up server and a subnet to connect a device to a remote host, the UDP/IP code is not limited to this type of connectivity. Variations to this implementation are possible with the correct setup. Application note AN2120/D describes a device, for example, that connects to an ISP, which then allows the device to reach a destination host across the Internet.

Demo Development Environment

As discussed above, the UDP/IP code can be modified and compiled within the CodeWarrior environment. To program the Avnet evaluation board, the P&E FLASH Programmer can be used.

The networking part of the demo was developed and tested using an IBM-compatible PC running these Microsoft® programs[1]:

- Windows® 98 SE
- FrontPage® 2000
- Internet Explorer 5.5
- Dial-Up Networking
- Dial-Up Server
- Winsock version 6 ActiveX component

The Windows networking setup described below should also be compatible with Windows 95, 98, NT®, and 2000.

Avnet Evaluation Board Setup

Before setting up and configuring the network for the demo, the Avnet board must be modified to be compatible with the application note AN2120/D hardware requirements. In addition, the Avnet board must also be configured with the correct clock and PLL settings for the demo by setting some jumpers and switches on the board.

1. Microsoft, Windows, Frontpage, ActiveX, and NT are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Avnet Evaluation Board Modifications

Modifications to the Avnet evaluation board are required so that the Avnet evaluation board will be wired as dictated by application note AN2120/D — see the serial port schematic **Figure 2** for the AN2120/D hardware requirements. This schematic can also be found on page 46 of the application note AN2120/D. These wiring requirements are a consequence of the coding of the UDP/IP protocol.

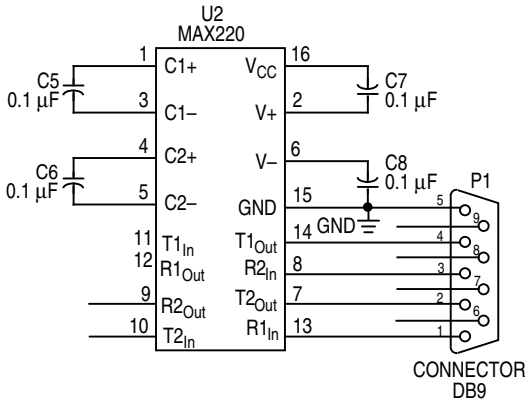


Figure 2. AN2120/D Serial Port Schematic

Comparing the AN2120/D schematic to the schematic for an Avnet evaluation board, it is evident that the Avnet board lacks two serial port hardware-wiring connections. (See **Figure 3** below. Please reference the Avnet ADS-MOT908GP32 for more detailed schematics.) **Table 1** summarizes the schematic observations. The table shows that the unmodified Avnet board lacks CD and DTR signals.

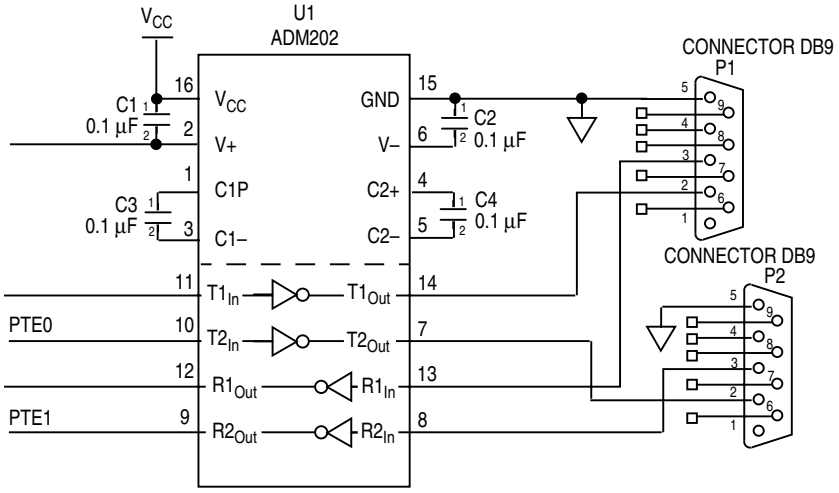


Figure 3. Avnet Serial Ports Schematic

Table 1. Serial Port Wiring

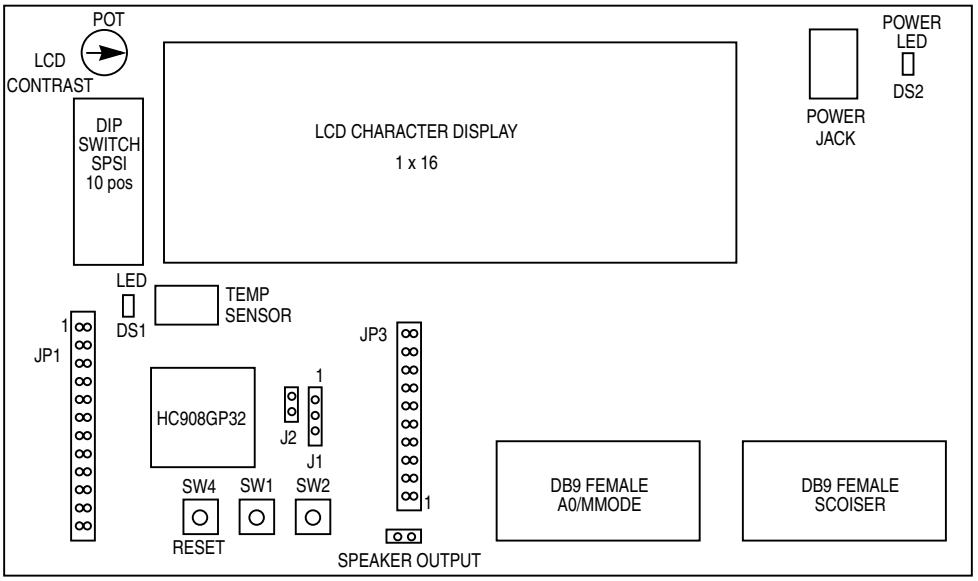
Pin	Pin Description	AN2120/D Requirements	Unmodified Avnet	Modified Avnet
3	Rx (= R2)	PTE1	PTE1	PTE1
2	Tx (= T2)	PTE0	PTE0	PTE0
1	CD (= R1)	PTD1	Lacking	PTD1
4	DTR (= T1)	PTD0	Lacking	PTA0

Table 1 also shows the required signal-to-port serial communication wiring modifying the Avnet board. The modifications are described in more detail below. Item 1 resolves the carrier detect (CD) wiring requirements; Item 2 resolves the data terminal ready (DTR) requirements. Layout schematics of an unmodified and a modified Avnet board are provided in Figure 4 and Figure 5, respectively.

Modifications:

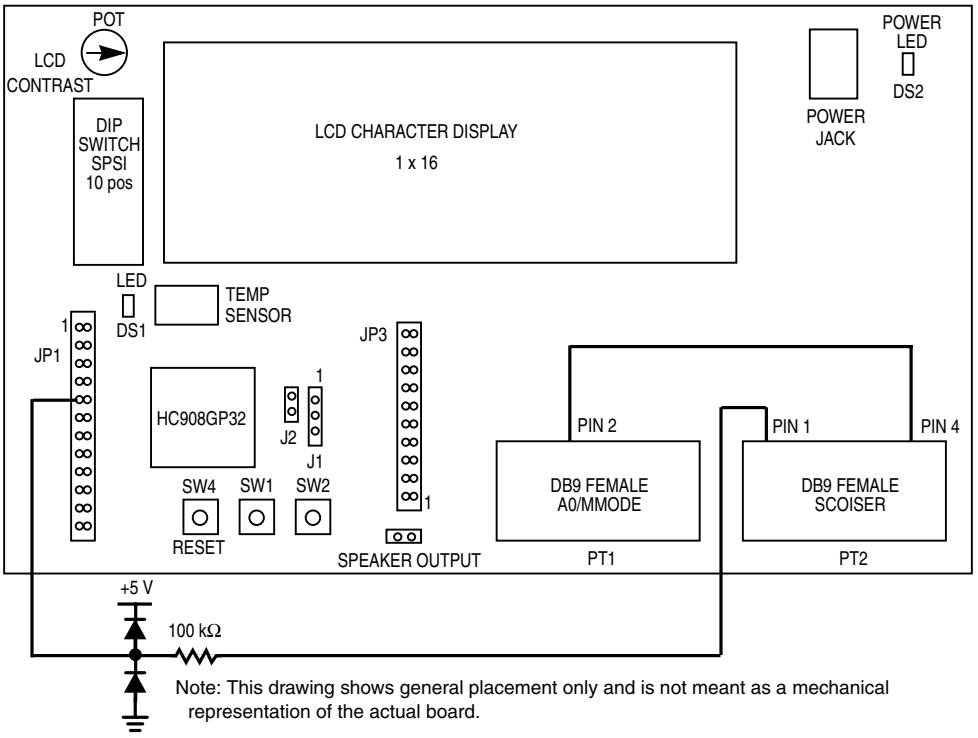
1. Jumpered user interface pin PTD1 (that is Avnet board JP1 pin 9) to resistor/diode clamp to the Avnet user serial port (PT2) pin 1
2. Jumpered the Avnet monitor mode serial port (PT1) pin 2 to the Avnet user serial port (PT2) pin 4

NOTE: *The item 2 modification, the DTR resolution, uses a different signal wire than expected by the UDP/IP code (PTA0 instead of PTD0) and will dictate a modification in the UDP/IP code. This modification is described in the section entitled* ***UDP/IP CodeWarrior Code Modifications****.*



Note: This drawing shows general placement only and is not meant as a mechanical representation of the actual board.

Figure 4. Unmodified Avnet Board Layout



Note: This drawing shows general placement only and is not meant as a mechanical representation of the actual board.

Figure 5. Modified Avnet Board Layout

Avnet Board Configuration

In addition to the hardware modifications, the Avnet board must be configured with the correct clock and PLL settings to operate with the UDP/IP code. Configuring the Avnet board requires changing the switches and jumpers. The Avnet board layout schematic (**Figure 5**) and the Avnet board mode configuration table (**Table 2**) can be used together to ensure the board is set up correctly.

The Avnet board must be configured in one mode for demo operation and a different mode when programming the Avnet board MCU. To configure for the demo, the Avnet board must be set up with the “User mode using the 32.768 kHz and internal PLL.” These settings are described in detail in the Avnet board documentation. To set up the Avnet board for programming, configure the Avnet board with the “Monitor mode 9600 baud” settings, as described below.

Table 2. Avnet Board Mode Settings

	MMode Bit 1	MMode Bit 2	Oscillator Divide	Oscillator Select	Jumper J1	Jumper J2(1)
Monitor mode 9600 baud(2)	On	On	On Osc/2	Off 4.9 M	2 and 3	Off
Monitor mode 4800 baud	On	On	Off Osc/4	Off 4.9 M	2 and 3	Off
Monitor mode 9600 baud using 32.768 kHz crystal (FLASH must be clear)	On	On	X	On-32 k	1 and 2	On
User mode using 32.768 kHz and internal PLL(3)	Off	Off	X	X	1 and 2	On
User mode using 4.9152 MHz oscillator(4)	Off	Off	X	X	2 and 3	Off

1. J2 must always be installed when using the 32.768 kHz crystal and removed when using the oscillator.
2. This setting was used to program the MCU on the Avnet board.
3. This setting was used for demo operation.
4. This is the normal operating mode for running the demos.

UDP/IP CodeWarrior Code Modifications

Several code modifications are required to successfully port the UDP/IP CodeWarrior code to the Avnet board. These modifications are primarily a result of the hardware modifications discussed above. Once these changes are made, the UDP/IP code can be compiled and the resulting S19 output file can be FLASH programmed onto the Avnet board with P&E FLASH Programmer or a similar tool. These modifications are detailed below.

Modifications to MCU_Setup.h:

These changes are required because of the serial port requirements of UDP/IP (see the section entitled **Avnet Evaluation Board Modifications**). While the Avnet board has two active pins on its user serial port, the UDP/IP expects a serial port with four active pins. In **Avnet Evaluation Board Modifications**, the hardware modifications that were made to the Avnet board to address this issue resulted in the hardware not matching what the software expected.

The issue is that while PTD1 was added to account for the missing serial port pins as expected by the UDP/IP code, PTA0 was substituted for PTD0. To resolve this hardware/software discrepancy, the UDP/IP code is modified to expect data/signals from PTA0, not PTD0. The code modifications that are required are shown here.

```
/*****************************************************************************
* Port Pin Setup for Modem
*****************************************************************************/
// Port A bit 0 is DTR
#define DTR_OUTPUT  DDRA  |= DDRA0;     // PTA0 set to output
#define DTR_ON      PTA   |=     PTA0;  // Set port pin
#define DTR_OFF     PTA   &= ~PTA0;     // Macro to set DTR OFF
#define DTR_PIN     (PTA & PTA0)        // DTR Pin = Pin 0 of PORT A
#define MODEMCDPOLARITYNORMAL FALSE     // CD polarity (true if inverting rs-232 buffer used)
```

Modifications to ModemDrv.c

The next UDP/IP code modifications are also a result of the hardware changes described in the section entitled **Avnet Evaluation Board Modifications**. In this case, a correction to the polarity of the CD line is required. The code changes are detailed here.

```
/*********************************************************************
Function : ModemOnLine
Parameters : None
Date : January 2001
Desc : Returns the status of the CD (carrier detect) signal.
*********************************************************************/
BYTE ModemOnLine (void) {
if (MODEMCDPOLARITYNORMAL)
          return (PTD & 0x02) ^ 0x02; // Return the status of the normal CD line
else
          return (PTD & 0x02);        // Return the status of the inverted CD line
}
```

Modifications to Variables.c

The third and last UDP/IP code modification depend on the particular setup of the network. For a specific network, the UDP/IP code needs to be changed to be compatible to the network's IP address and dial up phone number since IP addresses and phone numbers will undoubtedly change from implementation to implementation.

The `RemoteServer` variable describes the IP address of a destination host, while the `IPAddress` variable describes the IP address of the local device. In this case, the local device is the Avnet board. The `RemoteServer` and the `IPAddress` variables should be changed to be compatible with the network.

The `DIALNUMBER` variable, which describes the phone number of the ISP provider or dial-up server should be changed to the phone number of the ISP provider or dial-up server. In this case, the phone number is stored in the `ISP_phonenumber` variable.

```
/*********************************************************************
BYTE RemoteServer [4] = {10, 20, 5, 10};        // Remote Server to send notifications

BYTE IPAddress[4] = {10, 20, 5, 7};             // Default IP Address

const char * DIALNUMBER    = ISP_phonenumber;   // Dial out phone number

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

Device and Network Setup

Figure 6 illustrates how the devices must be connected for this demo.

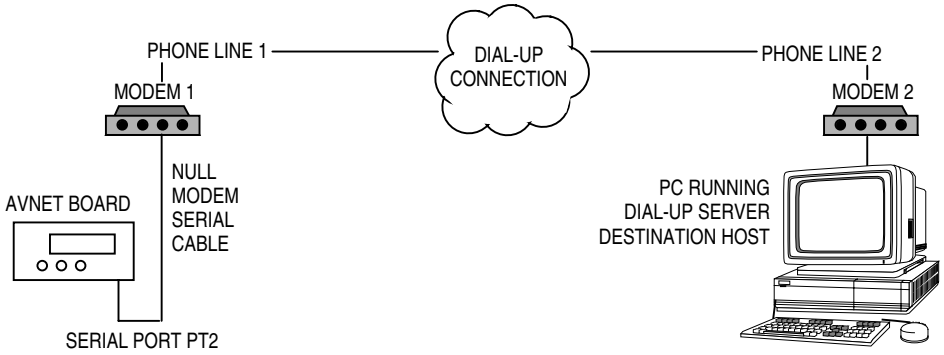


Figure 6. Demo Device Setup

Demo Components:

1. An IBM-compatible PC running Windows 98 SE is required for this demo. This PC must be configured with the components listed below. (The next section, entitled **Windows Dial-Up Server and Network Configuration**, gives a procedure to set up and configure each of the components listed below.)
   a. The PC must be set up as a dial-up server to accept incoming calls requesting a PPP connection.
   b. The PC must be set up with two TCP/IP devices. One device must be a dial-up adapter.
   c. The PC requires some type of mechanism to act as a destination host. (In this case, an HTML page embedded with a Winsock ActiveX component is used.)
2. A modified Avnet evaluation board running the UDP/IP CodeWarrior code that has been modified as described in **UDP/IP CodeWarrior Code Modifications** is required.
3. Two phone lines are required for this demo. As an alternative to two phone lines, a telephone simulator can be used.
4. Two Hayes-compatible modems:
   a. Modem1: An external modem is required to connect to the Avnet evaluation board. The interface between the Avnet evaluation board and the modem is a null modem serial cable. Modem1 is then connected to phone line 1. The Avnet evaluation board will dial out using phone line 1.
   b. Modem2: An external/internal modem is required on the PC running the dial-up server. Modem2 is connected to phone line 2 and must be configured to accept incoming phone calls. The phone number to phone line 2 must be known and programmed in to the Avnet board.

The first step is to connect the devices listed above as illustrated in **Figure 6**. Next, several Windows components must be configured. These include the Windows Dial-Up Server, Windows network protocol components, and the PC modem. The setup and configuration of these items is described in detail in the section entitled **Windows Dial-Up Server and Network Configuration**.

Windows Dial-Up Server and Network Configuration

This section describes how to set up the Windows network components of the demo. For the demo, a dial-up server running on a PC will act as the ISP. In addition to configuring the dial-up server, the other devices on the network (the Avnet board and the destination host) must be configured. All these network components will be configured to be on the same network subnet.

A detailed discussion is provided below for each of these items. Although the discussion is based on an IBM-compatible PC running Windows 98 SE, FrontPage 2000, Internet Explorer 5.5, Microsoft's Dial-up Networking Dial-up Server, and Microsoft's Winsock version 6 ActiveX component, different setup could also be used.

Setting Up the Dial-up Server

There are four steps to set up a PC to provide dial-up server capability. These steps are:

1. Configure your computer's modem
2. Install the Dial-Up Networking component
3. Install the Dial-Up Networking Server component
4. Enable and configure the dial-up server

This procedure is based on Article Q139710 from Microsoft's Technical Library. Please refer to Microsoft's Technical Library for more information.

Configuring Your Computer's Modem

For the Dial-Up Server, a modem and the modem drivers must be installed on the PC. If a modem is not installed, the modem can be installed following the procedure below.

Modem Installation Procedure:

1. Click **Start**, point to **Settings**, click **Control Panel**, and then double-click the **Modem** icon.
2. Follow the Modem Installation Wizard directions.

The modem also must be set up to connect only at one speed (9600 baud). This setting may be specified in the properties dialog box for the modem. As Figure 7 illustrates, to set this property, select a maximum speed of 9600 and check the **Only connect at this speed** checkbox.

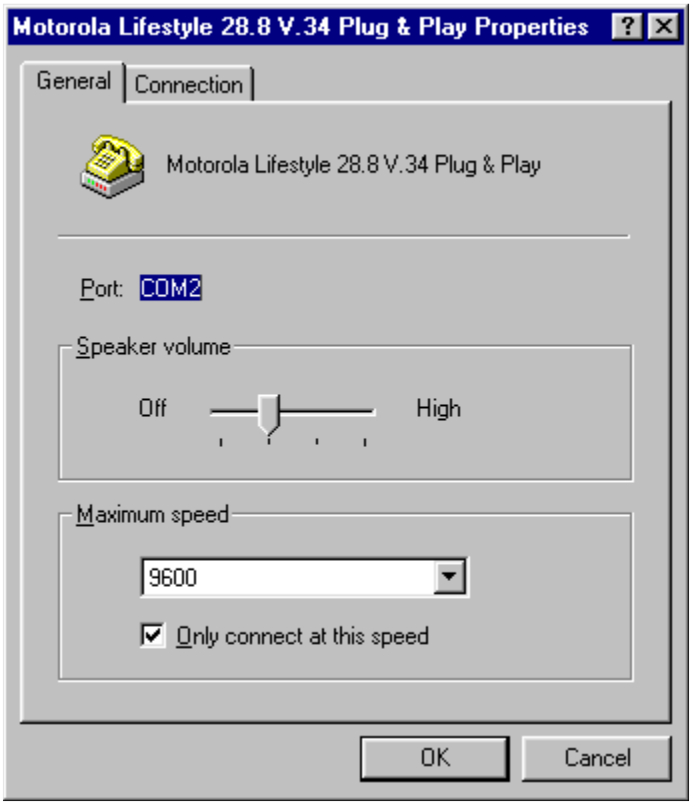


Figure 7. Modem Properties Dialog Box

Install the Dial-Up Networking and Dial-Up Networking Server Components

Next, both the Windows Dial-Up Networking component and Dial-Up Networking Server component must be installed on the PC (from the Windows installation disk). If these components are not already installed, a step-by-step installation procedure is provided below. **Figure 8** shows the dial-up installation options.

Dial-Up Networking Component Installation Procedure:

1. Click **Start**, point to **Settings**, click **Control Panel**, and then double-click the **Add/Remove Programs** icon.
2. On the **Windows Setup** tab, click **Communications** in the **Components** box, and then click **Details**.
3. Click to select the **Dial-Up Networking** check box.
4. Click to select the **Dial-Up Networking Server** or **Dial-Up Server** check box.
5. Click **OK**.

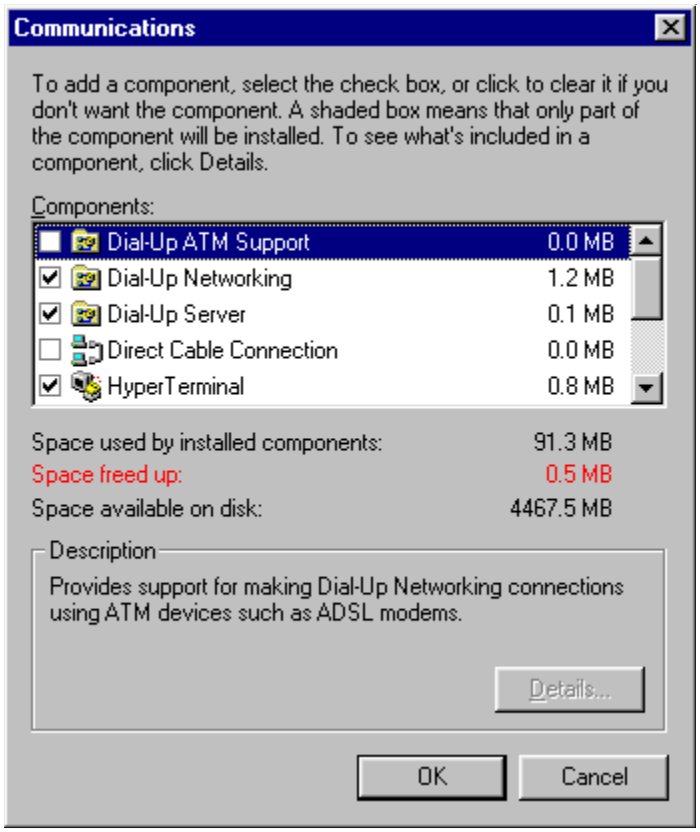


Figure 8. Communication Component Installation Dialog Box

Enable and Configure the Dial-Up Server

Once the dial-up components are installed, the dial-up server can be configured and enabled. To configure the dial-up server, open the dial-up server dialog box. To open the dial-up server dialog box, in **My Computer**, double-click the **Dial-Up Networking** folder. Then, on the **Connections** menu, click **Dial-Up Server** as shown in **Figure 9**.

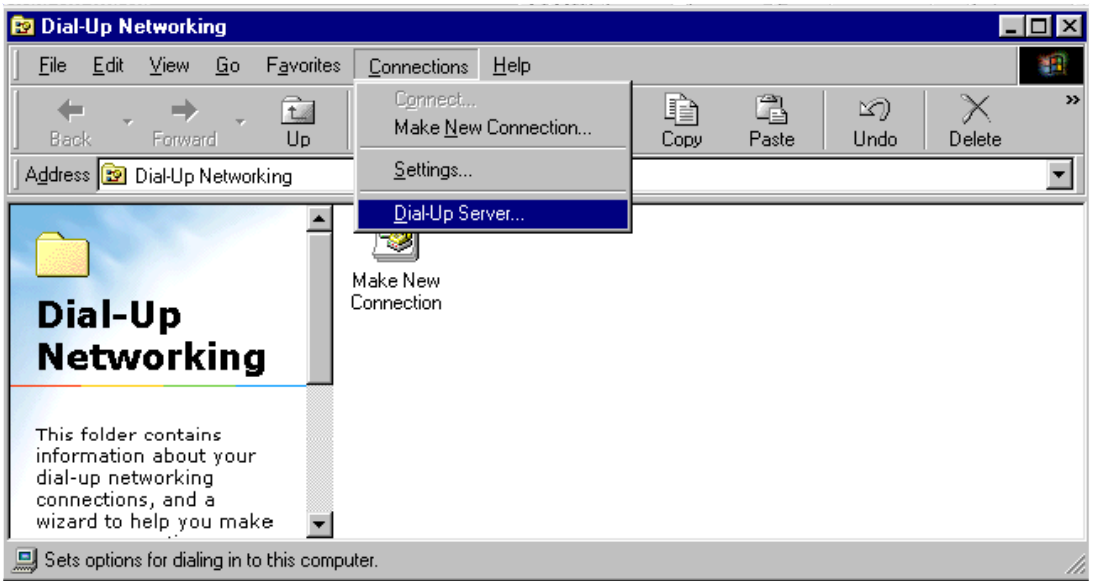


Figure 9. Dial-Up Networking Folder

With the dial-up server dialog box open, the dial-up server can be configured and enabled. First, enable the dial-up server by selecting the **Allow caller access** radio button (see **Figure 10**).

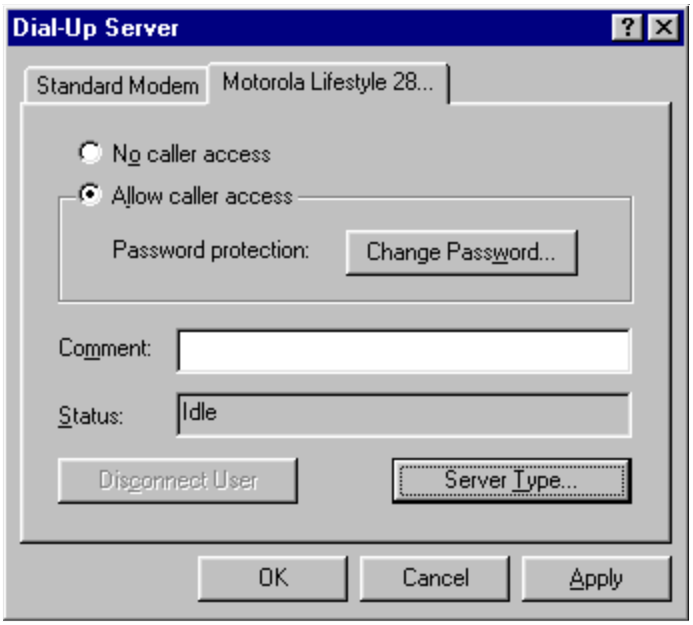


Figure 10. Dial-Up Server Dialog Box

The dial-up server type must now be configured, so click the **Server Type** button on the Dial-Up Server dialog box as shown in **Figure 10**.

In the server type dialog box, ensure that both the check boxes (see **Figure 11**) are unselected. In the drop-down list for selecting the type of dial-up server, select the **PPP: Internet, Windows NT Server, Windows 98** server option. Then, click the **OK** button to accept this configuration.

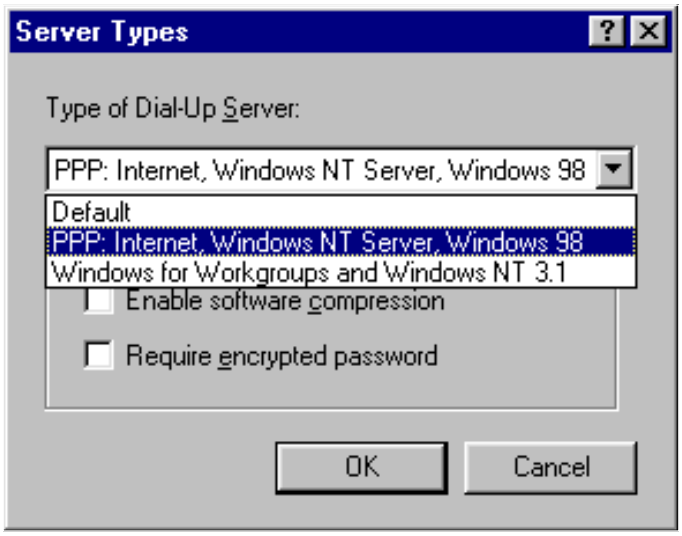


Figure 11. Server Types Dialog Box

The dial-up server dialog box should reappear. To enable the PC as a dial-up server, click the **Apply** then **Okay** buttons.

Setting up the Demo Network Subnet

Once the dial-up server is set up, configure a network for the demo. First, make sure the PC uses the correct network components. If the PC does not have the network components installed, the components must be added.

Adding Network Components

The PC must use these network components:

- Client for Microsoft Networks
- Dial-Up Adapter
- Two network protocols:
  - TCP/IP
  - NetBEUI
- File and print sharing

To ensure that the computer uses the correct network components, check the PC's network properties dialog box. Examine the list of installed network components, and, if any of the necessary components that are listed above are not installed, click **Add** to install them.

Accessing the PC Network Properties Dialog Box:

Click **Start**, point to **Settings**, and click **Control Panel**, and then double-click the **Network** icon.

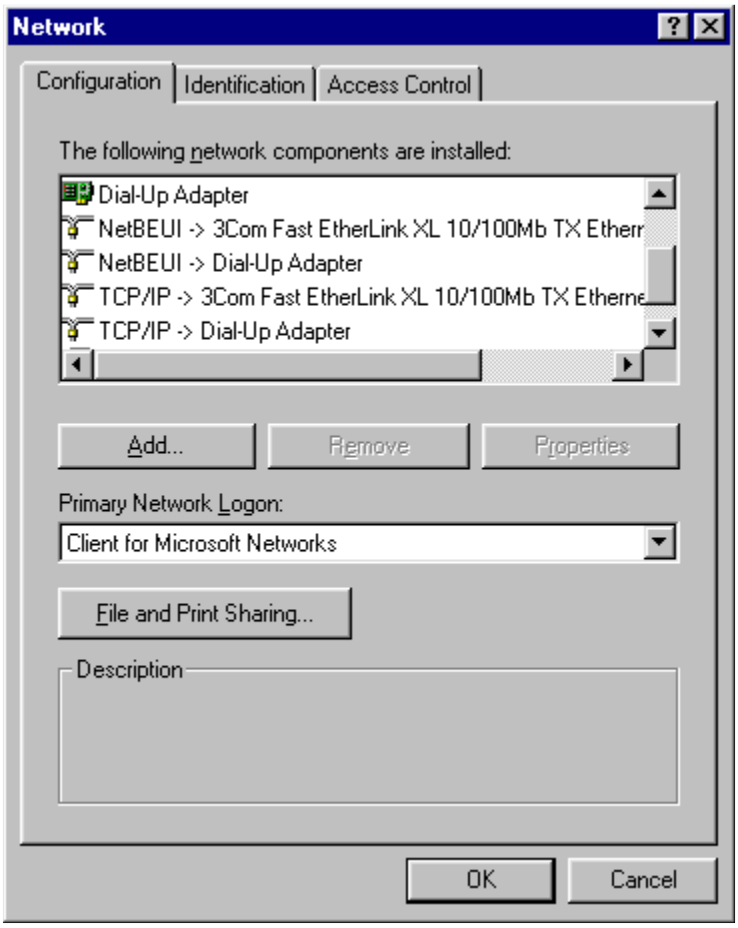


Figure 12. PC's Network Properties Dialog Box

Configuring the Network Protocol Components

As discussed above, the demo works with devices that are on the same network subnet.To set up the subnet for the demo, the IP address setting on the GP32 and the Dial-Up Server must be configured manually using non-routable IP addresses (i.e., 10.x.x.x, 90.0.0.x, 172.16.x.x through 172.32.x.x, or 192.168.x.x). In this particular setup, the devices were all configured with an IP address in the range 10.20.5.1 to 10.20.5.10. Recall that the UDP/IP code is programmed with an IP address of 10.20.5.10 for the Dial-Up Server and an IP address of 10.20.5.7 for the Avnet board. These IP settings and others need to be reflected in the configuration of the Dial-Up Server.

Configuring File and Print Sharing

For file and print sharing, the settings on the dial-up server PC must be configured as shown in **Figure 13**. To access this dialog, click **File and Print Sharing…** on the network properties dialog box (see **Figure 12**).

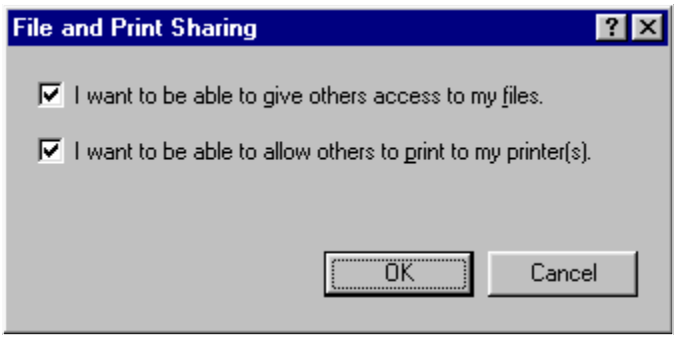


Figure 13. File and Print Sharing Dialog

Configuring the Dial-Up Server TCP/IP Protocol

To set up the IP address on the dial-up server, select the TCP/IP protocol network component for the dial-up adapter (see **Figure 12**) and click **Properties**. With the TCP/IP dialog box open, as shown in **Figure 14**, select the **Specify an IP address** radio button. A specific IP address can now be entered manually as well as a subnet mask. These values are provided in **Table 3**.

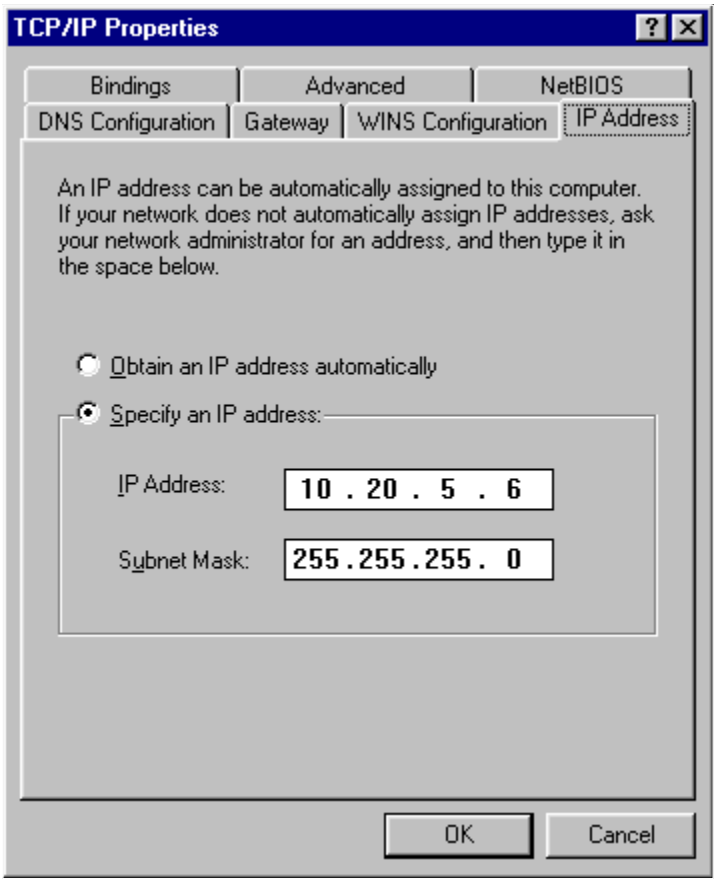


Figure 14. TCP/IP Network Protocol Property Dialog Box

Table 3 below indicates the specific IP address used in this demo. The TCP/IP address must be set up not only for the dial-up adapter, but for also a second TCP/IP device. The second TCP/IP device in this demo is the PC's LAN card. The second TCP/IP device is needed because it acts as the destination host. As the destination host, it will receive all broadcasts from the Avnet GP32 evaluation board.

Table 3. TCP/IP Setting for the Dial-Up Server

TCP/IP Device	IP Address	Subnet Mask	Comments
TCP/IP: dial-up adapter	10.20.5.6	255.255.255.0	ISP IP address (Dial-Up Server)
TCP/IP: LAN card	10.20.5.10	255.255.255.0	Destination Host IP address

Configuring the GP32 Network TCP/IP Protocol

To set up the Avnet evaluation board to dial the dial-up server and to access the same subnet as the dial-up server, the IP addresses in the Avnet evaluation board CodeWarrior code must have the same IP address prefix as the dial-up server. In this case, the subnet IP address prefix is 10.20.5.x.

The Avnet evaluation board CodeWarrior code has two IP address settings that must be set to make the device compatible with the dial-up server subnet. These two IP address settings were detailed in **UDP/IP CodeWarrior Code Modifications**. For completeness, these settings are repeated in **Table 4**.

Table 4. Avnet Board Code Variable IP Address Settings

TCP/IP Device	IP Address	Subnet Mask	Comments
Remote Server IP address	10.20.5.10	255.255.255.0	Destination Host IP address
Local IP address	10.20.5.7	255.255.255.0	Avnet GP32 evaluation board IP address

With the IP address configuration described above, the devices will be configured to be on the same network subnet and, therefore, the devices are guaranteed to have connectivity among them.

Setting Up the Windows Winsock ActiveX Demo

An HTML page is needed for the demo. The HTML page is intended to provide an interface for the destination host to receive broadcasts from the Avnet board across the network.

The HTML page is embedded with an ActiveX component that is controlled with VBScripting. Specifically, the HTML page contains Microsoft's Winsock Control (version 6). For the demo, the Winsock control is configured/programmed to monitor data from the Avnet board IP address. The properties, methods, and events of the Winsock Control are manipulated using VBScripting. Besides the ActiveX Winsock Control, the HTML page also contains other standard components like buttons and textboxes that allow both user input and data displays.

To operate the demo, open the HTML page. The page's functionality is very simple. It echoes the broadcast, "Greetings from the HC08," that it receives via the PPP/UDP link from the Avnet board. Since the UDP/IP is coded only to broadcast the message "Greetings from the HC08," to have a more applications-related demo, the UDP/IP code must be modified.

Connecting the Avnet to the Dial-Up Server

With all the components configured and connected as described in the section above, the demo is ready for network communication. Confirm the devices on the network are connected as described in **Device and Network Setup**, and then power-up (or reset) the Avnet board. The Avnet board will start executing the UDP/IP code on its HC908GP32 microcontroller. The Avnet board will then dial the dial-up server and establish a PPP connection to the dial-up server. At the same time, the HTML page for the destination host can begin capturing the data coming in from the network. The demo shows how the UDP/IP CodeWarrior version of the code works with a modified Avnet board (with an HC08 MCU) to connect to an ISP using a PPP connection.

Debugging a Network Connection

Debugging the network connection described by application note AN2120/D can be challenging. **Figure 15** gives an overview of the network and its possible debugging targets. There are two main areas where the debugging efforts are focused:

- Debugging the code being executed on the Avnet board MCU
- Debugging the Avnet board network connection to the ISP (Dial-Up Server)

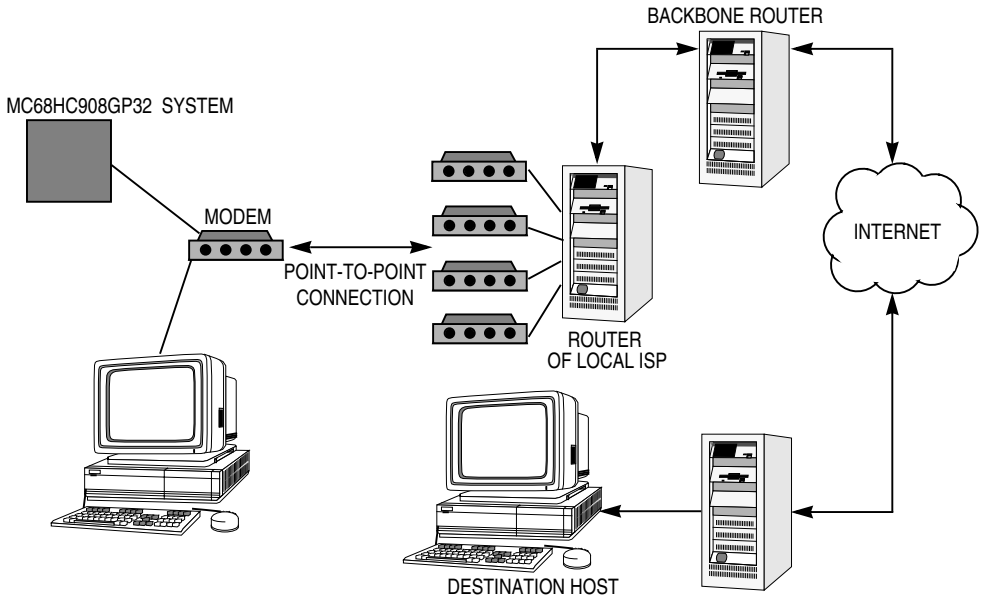


Figure 15. Overview of Network Device to Debug

In both of these instances, the debugging obstacle is the lack of visibility within the MCU and the network connection. The information which follows describes a methodology and tools that can be useful when debugging this type of system.

Debugging the MCU

MMDS Development Environment

To debug an MCU embedded in a device like the Avnet board, the preferred debugging setup is using an MMDS development tool with a UART to provide the serial connection. An Avnet board with an HC908GP32 socket is required in this setup to provide an interface for the MMDS. With this setup, the Avnet board can be connected as described above and the MMDS can be used to provide the in-circuit debugging capability.

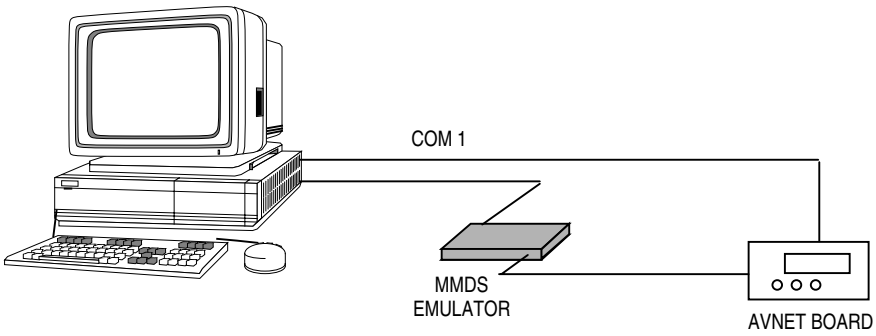


Figure 16. MMDS Debugging Setup

Alternative MCU Debugging

If an MMDS tool is not available, an alternative debugging tool or method can be employed to debug the MCU. The method described below illustrates one possible solution for an alternative MCU debugging methodology. The methodology presented below uses the transmit procedure that is a part of the UDP/IP code.

The transmit procedure takes a string as an argument and then transmits the string to the Avnet board's serial port via the MCU's serial communication interface (SCI). As the UDP/IP code is written, it currently transmits modem commands and UDP packets to the SCI.

To provide a debugging mechanism, write debug messages to the serial port with the transmit procedure. The debug messages will then be transmitted via the SCI when they are encountered in the code. For this alternative MCU debugging methodology to be successful, the debug messages must be filtered out of the serial port data stream. These debug messages will become non-intrusive breakpoints since the debug messages will indicate what part of the code is being executed.

As discussed above, as the code is being executed on the Avnet board, all messages transmitted via the SCI, including the debug messages, are sent to the modem through the serial port on the Avnet board. For the method being described, instead of the serial data going straight to the modem, the serial port data stream must be sent through a filter to extract and display the debug messages. To filter the data stream, the serial port data between the modem and the Avnet board must be diverted and accessed (see **Figure 17**). The serial data link must be broken and then linked back together with a third device, a PC for example. The third device must filter out and display the debug messages while sending only the appropriate messages to the modem. Being able to analyze the debug messages provides an alternative for the debugging of the MCU.

MCU Debugging Features Summary

Debugging of the MCU can be approached with different tools. All of these approaches, like the ones described above, have common features that are necessary for debugging an MCU. A summary of these features is provided here:

- In-circuit debugging capability
- Real-time debugging
- Serial port data filtering
- MCU program breakpoint capability
- Debug message display

Debugging the Serial Port

To debug the serial connection of the Avnet board to the modem, a serial port communications protocol analyzer tool is required. This tool should be able to monitor the serial communications port, its protocol in use, and all serial port traffic. This serial port protocol analyzer can take the form of an ActiveX program or a commercial software package.

In this application, because two serial devices (the Avnet board and the modem) are used to establish a dial-up connection, the serial communication activity between both devices must to be analyzed. One method to provide a serial port data monitor between the two devices is to first divert the serial data stream to a serial protocol analyzer, non-intrusively monitor the serial communication, and finally direct the serial data to its original destination.

To implement this strategy, the serial data link must be broken and then linked back together with a third device that has two available serial ports, a PC for example. To link the PC serial ports together, an application using Microsoft's MSCOMM32 ActiveX object or similar tool can be employed. **Figure 17** illustrates a typical application of the technique described here. At the same time, the PC must analyze the serial data stream providing detailed information about the serial data protocol as well as timing information, error flags, and modem status flags.

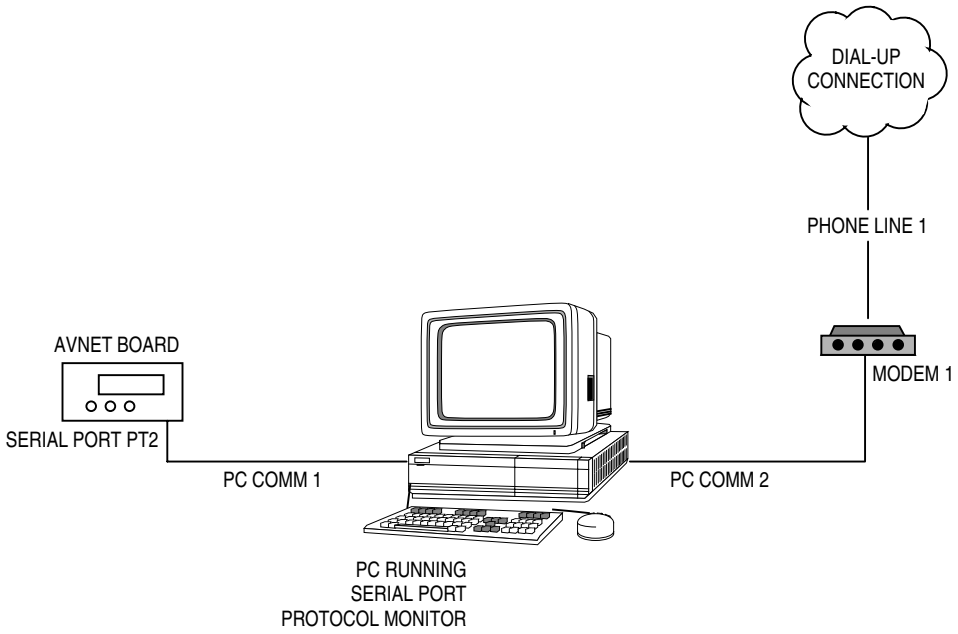


Figure 17. Serial Data Stream Monitor Tool Configuration

Serial Port Debugging Features Summary

The ideas for debugging the serial port presented above describe what features are necessary to resolve serial port issues. A summary of the features needed for serial port debugging is listed here:

- Enabled serial ports
- Real-time serial port data monitoring
- Serial port modem status
- Serial port line status
- Serial port timing details
- Serial port baud information
- Serial port analyzer data and session recording capability
- Serial port linking capability

Debugging the Network Connection

Issues with the network connectivity typically result in an error in the network setup and configuration of either the server (ISP) or the remote devices. Two possible network issues are possible. The first is that the network connection cannot be established. The other network issue encountered is that the network connection is established but the devices on the network are not talking to each other. Both of these issues can be debugged with the approach documented in the following sections.

Network Setup Debugging

This debugging discussion deals with the issue of the network not establishing a connection. The key steps to resolve this type of network bug is to determine how the network is designed and how the remote device must be configured to accept a connection.

Debugging the network, in this case, requires an understanding of how the network is implemented and in what configuration is the network functional. In addition, the requirements of the remote device must be considered. The remote devices must be compliant with the network's design structure and protocols. When the network is set up and configured correctly, the devices will connect.

Defining, Setting Up, and Configuring the Network

With an understanding of the network design and its connection capabilities, its network restrictions, and its underlying communication protocols (i.e., TCP/IP and NETBEUI), a user can configure the network and the remote devices. The configuration process of the network and remote devices for this demo illustrates this. An overview of the configuration process is included below to provide a guide to define and set up a network.

Network Setup Guide

In the implementation of UDP/IP connection for this demo, the network designed is a subnet. The demo subnet consists of a remote device, a dial-up server, and a destination host. For a subnet, all the devices must be configured with the common IP address prefix (10.20.5.x). More detail on the subnet configuration is provided in the section entitled **Windows Dial-Up Server and Network Configuration**. This configuration includes setting and configuring IP addresses, modem settings, network components, network protocols, server type, and other parameters.

If a UDP/IP connection is implemented with a different network configuration, using an ISP instead of a dial-up server for example, the correct network setting must be implemented. The guide above applies to other network setup variations as well. Following this guide will ensure that the network is defined and set up correctly.

Network Connectivity Debugging

Once the device can establish a connection, the network may still require tweaking. The primary problem that arises is that the devices are unreachable or the network connection is misdirected. In that case, the network IP address, network port information, modem setting, network components, network protocols, and server type must be reviewed, and, with this information, the server (ISP) or the remote devices can be tweaked as necessary as described in the section entitled **Defining, Setting Up, and Configuring the Network**.

Windows Network Analysis Tools

Windows provides many tools that are used to confirm the connectivity of devices on the network, *Ping* for example. Since the Ping functionality is included in the UDP/IP code, it can be used to debug the network connection. If the command confirms a valid connection to a remote device, then the network is up and running. If, on the other hand, the Windows command does not confirm a network connection to the remote device, then the network is not configured correctly. The next step in debugging the network is to enable more visibility of the network connection. This requires a network protocol analyzer. A discussion on network protocol analyzer is provided in the next section.

Network Protocol Analyzer

A network protocol analyzer is used to monitor the connectivity activity of the Internet or a local area network (LAN). The tool is invaluable for debugging network connection problems. The tool is capable of non-intrusively attaching itself and monitoring a dial-up connection or an Ethernet card. The network protocol analyzer can take the form of an in-house or commercial software package.

The overriding feature of the network protocol analyzer is its ability to capture, analyze, and decode network packets. Furthermore, the network protocol analyzer must be capable of determining the communication protocol of the network data packets. In addition, the program must be able to display a list of network connections, the IP addresses of the connections, the data direction, and the network data port information. The network protocol analyzer provides the detailed network information needed to debug a network.

Remote Device Debugging

Although the sections above have discussed the importance of a correctly configured remote server, these techniques can be used to debug remote devices. The bottom line is that the remote devices must be compliant with the structure of the network to which they are connecting. For the demo, for example, the two remote devices (the Avnet board and the destination host) must have the same IP address prefix (10.20.5.x for example).

Network Debugging Features Summary

Debugging a network connection is challenging but possible with a systematic methodology and the correct tools. The right features in a tool can streamline network debugging. A summary of these features is provided here:

- Capture network traffic from both an Ethernet card and a dial-up adapter
- View IP connection statistics like IP addresses and ports for both the server (ISP) and the remote devices
- Capture and decode network packets in real time
- Ping and network trace tools

Debugging Tool Examples

In debugging this application, several commercial and user developed tools were used. These tools and methods were used separately and combined, in the development of the demo, to assist in the debugging of the Internet connection. These tools have been itemized in Table 5.

Table 5. Debugging Tool Examples

Debugging Target	Tools Utilized
MCU	• Displaying debug messages with UDP/IP transmit procedure • User developed HTML page with ActiveX (MSCOMM32.ocx)
Serial Port	• User developed HTML page with ActiveX (MSCOMM32.ocx) • Commlite32, Real-time Control, Inc.
Network Connection	• Windows Ping & NetStat, Microsoft • CommView, TamoSoft, Inc.

Although these methods and tools were employed to develop and debug this demo, Motorola does not recommend or endorse any particular methodology, tool, or vendor. These tools are only provided to describe the generic principles and features that may be required to debug an Internet connection.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://www.motorola.com/semiconductors

MOTOROLA