SENA
TECHNOLOGIES

HD132x Series User Manual

Version 1.1

HD132x Series User Manual

Version 1.1

Last revised on May 28, 2001

Printed in Korea

Copyright

Sena Technologies is having the right to make changes and improvements to its product without providing notice.

Trademark

HelloDevice™ is a trademark of Sena Technologies, Inc.

Windows® is a registered trademark of Microsoft Corporation.

Ethernet® is a registered trademark of XEROX Corporation.

Notice to Users

When a system failure may cause serious consequences, protecting life and property against such consequences, with a backup system or safety device are essential. The user agrees that protection against consequences resulting from system failure is the user's responsibility.

This device is not approved for life-support or medical systems.

Changes or modifications to this device not explicitly approved by Sena Technologies will void the user's authority to operate this device.

Company Address

Sena Technologies, Inc.

210 Yangjae-dong, Seocho-gu

Seoul 137-130

Korea

Phone: +82-2-573-7772

Fax: +82-2-573-7710

Email: info@sena.com

URL: http://www.sena.com/

Contents

1. Before You Start

▶ Items supplied in each model of the HD132x Series

Items		HD1320E	HD1320	HD1321
Hardware	**Product Type**	External Box	PCB Board	PCB Board
	Power Supply Adaptor	❍	-	-
	RS232 Cable	❍	-	-
Software	**CD-ROM**	❍	-	-
		Download the latest version of HelloDevice Utility Software and Demo Sample Programs at http://www.sena.com.		
User Manual Hardcopy		❍	-	-

▶ Operating Environment

- Voltage Supply = 7.5V ~ 9V DC (HD1320/1320E), 5V DC ±10% (HD1321)
- Current Supply = 150mA min.
- Operating Humidity = 0 ~ 95%
- Operating Temperature = 5°C ~ 55°C

▶ Technical Support

Sena Technologies, Inc.
210 Yangjae-dong, Seocho-gu
Seoul 137-130
Korea
Tel: (+82-2) 573-7772
Fax: (+82-2) 573-7710
E-Mail: support@sena.com
Website: http://www.sena.com

2. Overview

HD132x Series is an RS232 to TCP/IP protocol converter, which enables data communication between the device and the user, over the Internet as client and/or server. With HD132x Series, users can easily send data to the device and/or receive data from the device over the Internet.

The HD132x Series is divided into three models depending on the product type:

- External box (HD1320E)
- Board (HD1320)
- Module (HD1321)

An existing device with the serial port (RS232) can be connected to the Internet simply with an external type HD1320E without using any expensive dedicated PC and lengthy serial cables.

Manufacturers can easily integrate their own network-enabled products with a board type HD1320 or module type HD1321 into the circuit board of the product without major design changes.

Table 2.1 shows the basic specifications of the HD132x Series.

	HD1320E	HD1320	HD1321
Processor	Ubicom SX52BD, 50 MIPS		
Memory	32KB SRAM, Buffering space for incoming data from the serial port		
Ethernet Interface	RJ45 Ethernet Connector		10BaseT Ethernet MAC & PHY Interface
Serial Interface	Male DB9 RS-232 Serial Port		TTL RS232 Serial Port
Supported Protocols	ARP, IP/ICMP TCP/IP Server & Client DHCP Client		
Management	HelloDevice Utility Software (for Windows 95/98/NT/2000 platform)		
Physical Properties	98mmx66mmx30mm 108g	86mmx53mmx16mm 44g	41mmx38mmx16mm 10g

Table 2.1 Basic Specifications of HD132x Series

3. Product Specifications
3.1 HD1320E / HD1320

The hardware configuration of HD1320E and HD1320 are shown below in Figure 3.1(a), (b), and, the system block diagram in Figure 3.2.

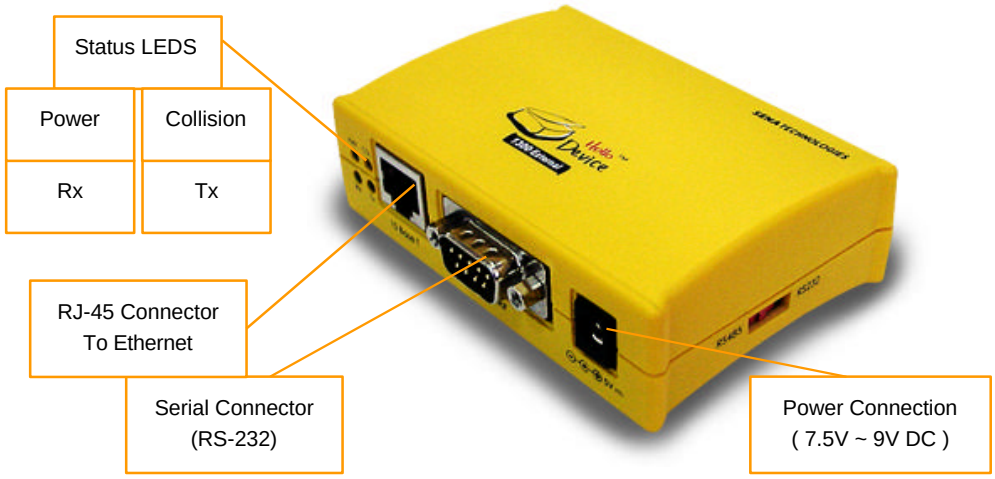


Figure 3.1 (a) HD1320E

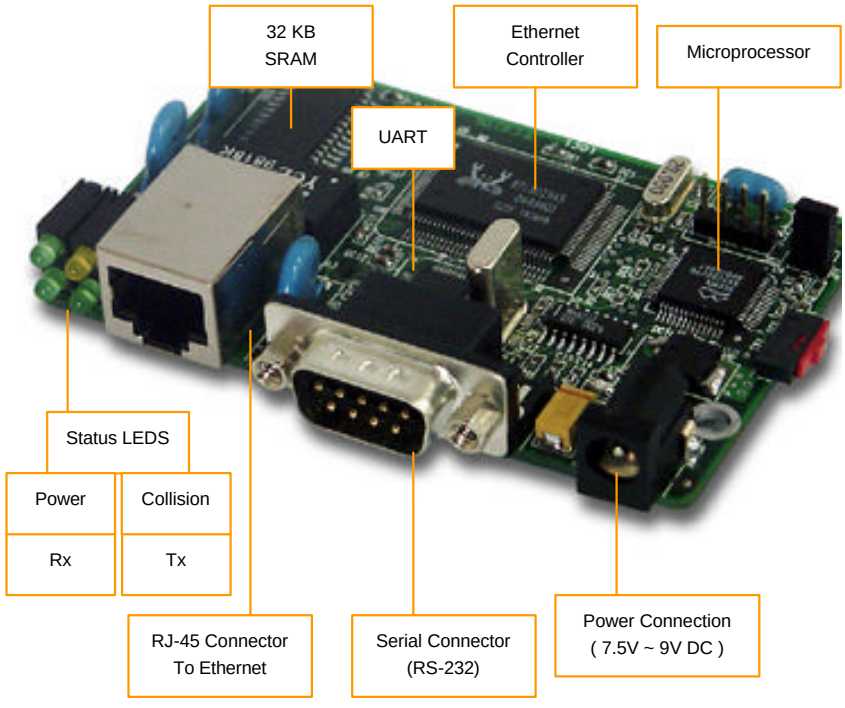


Figure3.1 (b) HD1320

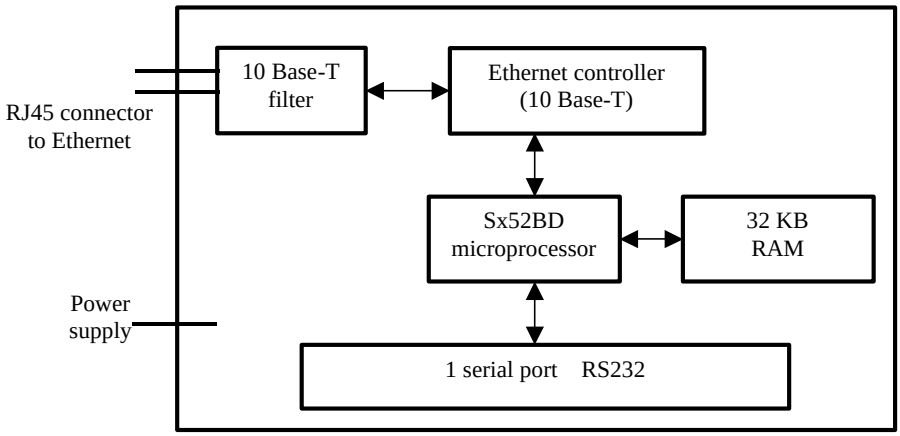


Figure 3.2 HD1320 System Block Diagram

3.1.1 Power Supply

- Voltage Supply = 7.5V ~ 9V DC
- Current Supply = 150mA min.

3.1.2 Ethernet Interface

The HelloDevice is directly connected to an Ethernet hub or switch by the RJ45 connector. Distances of up to 100m are supported.

(1) Ethernet Controller

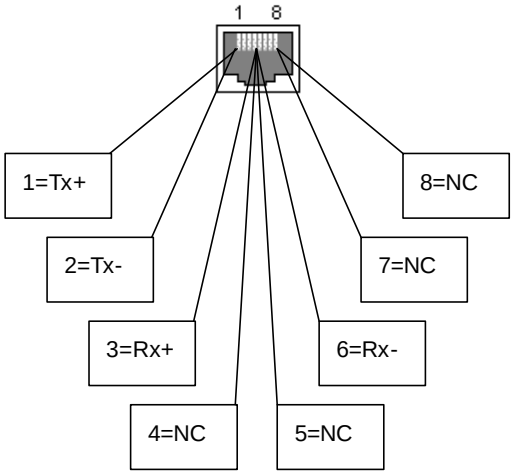
The Ethernet controller provides functions of framing, addressing data, detecting, avoiding errors and collisions.

Specifications:

- RealTek Full-duplex Ethernet Controller : RTL8019AS
- IEEE802.3 10Base-T Support
- NE2000 Compatible

(2) RJ45 Connector

- Shielded Connector compliant with AT&T258 specifications



Pin No.	Description	Wire Color
1	Tx+	White with orange
2	Tx-	Orange
3	Rx+	White with green
4	Not used	Blue
5	Not used	White with blue
6	Rx-	Green
7	Not used	White with brown
8	Not used	Brown

Figure 3.3 RJ45 Connector

(3) Status Indicator LED's

There are four status indicator LED's to indicate the following: (See Figure 3.1)

- **Power LED**

  Indicates the Power-On status of the HelloDevice.

- **Rx LED**

  Indicates that the Ethernet controller is receiving Ethernet packets from the network or not. If it is correctly connected to the network, this LED will blink when packets are received

- **Tx LED**

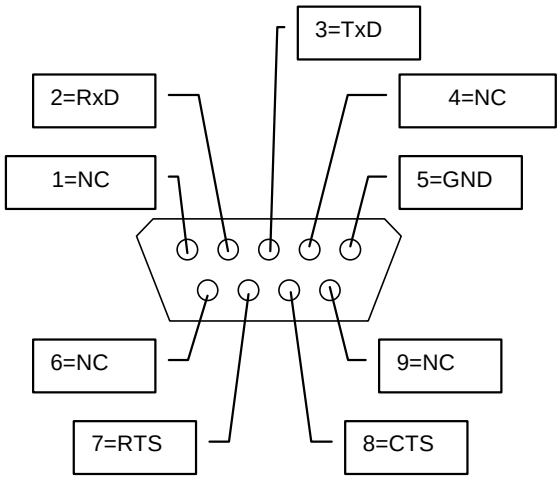
  Indicates that the Ethernet controller is sending packets. Before setting the HelloDevice network configuration, this LED will blink for every second. After network configuration, if connected correctly, this LED blinks only in response to requests from the user's PC, for example ping or server activity.

- **Collision LED**

  Indicates that one or more Ethernet packets transmitted from the Ethernet controller to the network may have collided with another packets. In this case, the Ethernet controller will automatically resend the packets.

3.1.3 RS232 Communication Interface

- DB9 Connector for RS232 communication
- Serial speeds 150bps ~ 115Kbps



Pin No.	Signals	Descriptions	Functions
1	NC	Not connected	-
2	RxD	Receive Data	Data Input
3	TxD	Send Data	Data Output
4	NC	Not connected	-
5	GND	Signal Ground	-
6	NC	Not connected	-
7	RTS	Ready To Send	Handshake Output (+12V=Ready to Rx, -12V=Not Ready to Rx)
8	CTS	Clear To Send	Data sent
9	NC	Not connected	Data sent only at +3V ~ 12V

Figure 3.4 Pin Assignment of DSUB Connector

3.2 HD1321

The HD1321 hardware structure is shown in Figure 3.5 and the system block diagram is as shown in Figure 3.6.

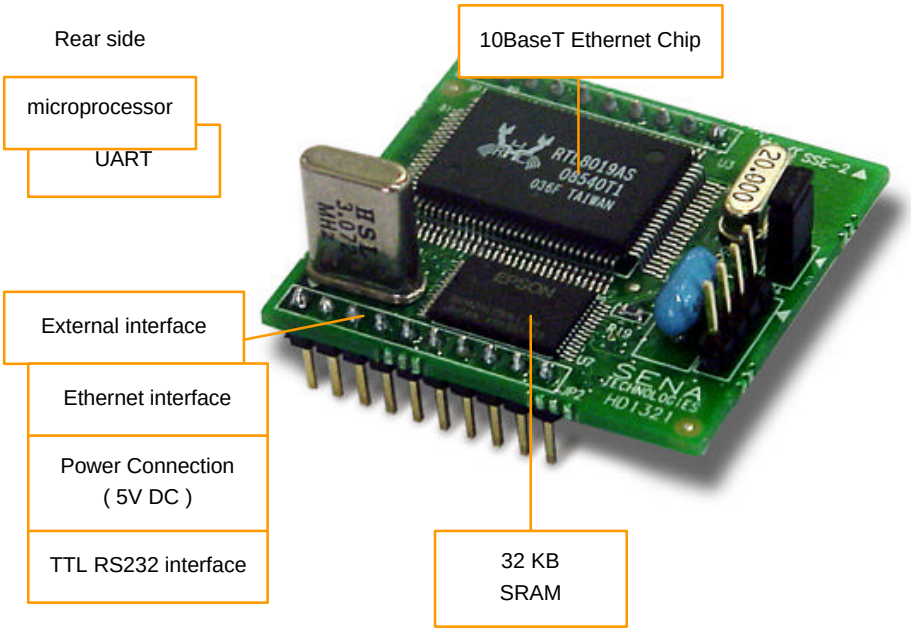


Figure 3.5 HD1321 Hardware Structure and Names

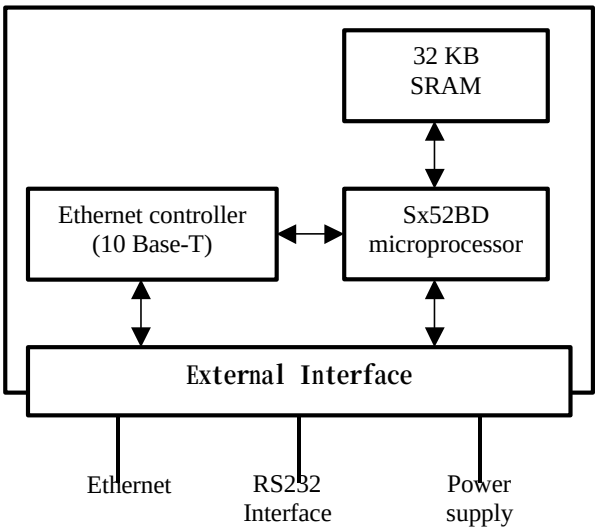


Figure 3.6 HD1321 System Block Diagram

3.2.1 Power Supply

- Voltage Supply = 5V DC ±10%
- Current Supply = 150mA min.

3.2.2 External Interface

There is a total of 20 pins in the connector JP1 and JP2 in HD1321 to interface microprocessor, Ethernet controller and UART with the user device. (See Table 3.1 and Figure 3.7)

Connector	Pin #	Symbols	Descriptions	Remarks
JP1	1	GND	Power Supply Grounding	
	2	Reset	System Reset	Low Active
	3 ~ 5	LED2 ~ LED0	Ethernet Communication Status LED LED0: Tx, LED1: Rx, LED2: Collision	
	6	TPOut-	Ethernet Signal TPOut-	
	7	TPOut+	Ethernet Signal TPOut+	
	8	TPIn+	Ethernet Signal TPIn+	
	9	TPIn-	Ethernet Signal TPIn-	
	10	Vcc	5V Power Supply	
JP2	1	Vcc	5V Power Supply	
	2	NC		TTL Level
	3	DSR	RS232 Data Set Ready	
	4	CTS	RS232 Clear To Send	
	5	DTR	RS232 Data Terminal Ready	
	6	RTS	RS232 Ready To Send	
	7	RxD	RS232 Data Input	
	8	TxD	RS232 Data Output	
	9	NC		
	10	GND	Grounding	

Table 3.1 HD1321 Pin Assignment

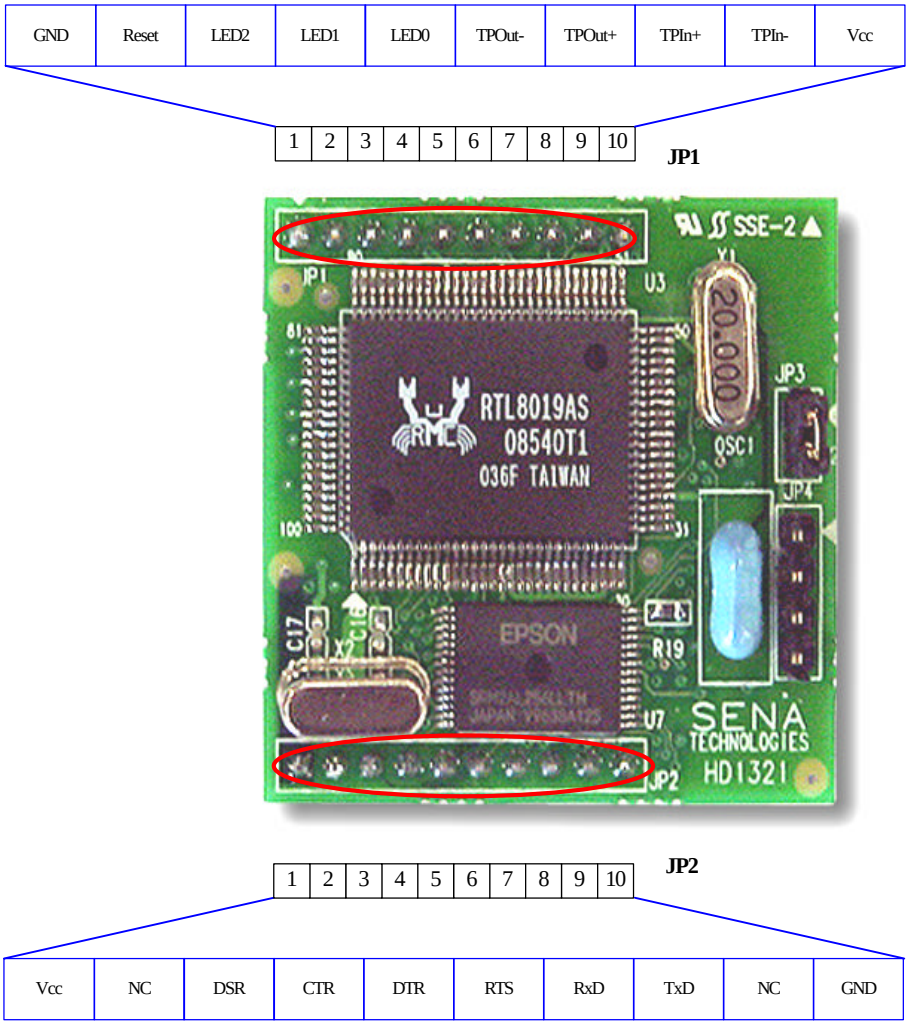


Figure 3.7 HD1321 External Interface

3.3 Firmware Structure

The HelloDevice Firmware resides in the program memory of microprocessor. The firmware includes a TCP/IP stack conforming to the Open System Interconnection (OSI) requirements for an Internet connection, and the serial communications interface for connection to the user device.

<table>
<tr><th colspan="2">Layer</th><th colspan="4">HelloDevice Support</th></tr>
<tr><td>7</td><td>Application</td><td>RS232 Communication Interface</td><td>System Management</td><td>DHCP</td><td></td></tr>
<tr><td>6</td><td>Presentation</td><td></td><td colspan="2"></td><td></td></tr>
<tr><td>5</td><td>Session</td><td></td><td colspan="2"></td><td></td></tr>
<tr><td>4</td><td>Transport</td><td></td><td colspan="2"></td><td></td></tr>
<tr><td>3</td><td>Network</td><td>TCP</td><td colspan="2">UDP</td><td></td></tr>
<tr><td>2</td><td>Data link</td><td colspan="3">IP / ICMP</td><td>ARP</td></tr>
<tr><td>1</td><td>Physical layer</td><td colspan="4">Ethernet (IEEE802.3)</td></tr>
</table>

Table 3.2 OSI 7 Layers and HD132x Firmware

4. Setting Up Your HelloDevice

The HelloDevice is installed in a following order:

1. Connect the HelloDevice to the power supply and Ethernet
2. Install the HelloDevice Utility Software on your PC
3. Assign an IP address of the HelloDevice
4. Set parameters of the HelloDevice

4.1 Connecting Hardware

4.1.1 Connecting HD1320 / 1320E

Since the hardware architecture of HD1320E and HD1320 is identical to each other, so is the setup process. In this section, HD1320E is used as an example.

(1) Connect the 7.5V ~ 9V DC power supply adaptor to HD1320E.

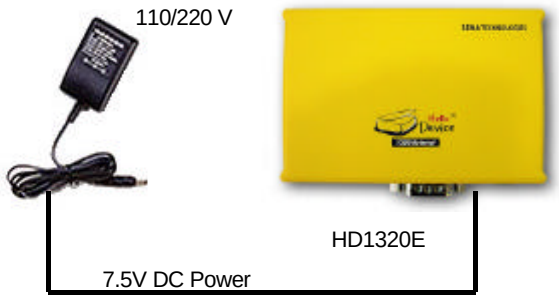


Figure 4.1 Power Connection to HD1320E

(2) Connect HD1320E to the Ethernet hub through RJ45. (Refer to Chapter 3 for details on RJ45 connector.)

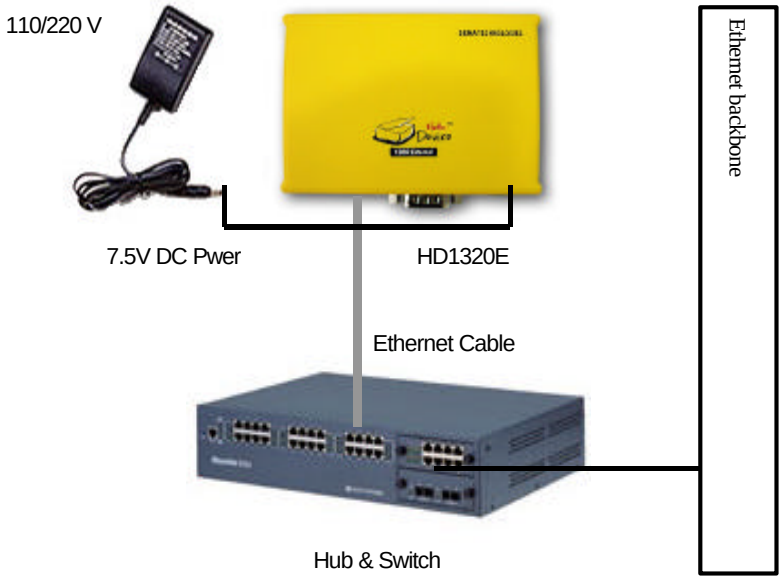


Figure 4.2. Connection of HD1320E and Ethernet Cable

(3) Check to see if the Tx LED in Status LED's blinks every second. Tx LED blinks periodically when an IP address is not assigned, which is normal at this stage. (See [Status LED's] in Figure 3.1).

4.1.2. Connecting HD1321

HD1321 is interfaced with the user device through the DIP type connector, and the power supply and RJ45 connector are implemented on the user device. Figure 4.6 shows the pin assignment of the connector and the interface circuit diagram.

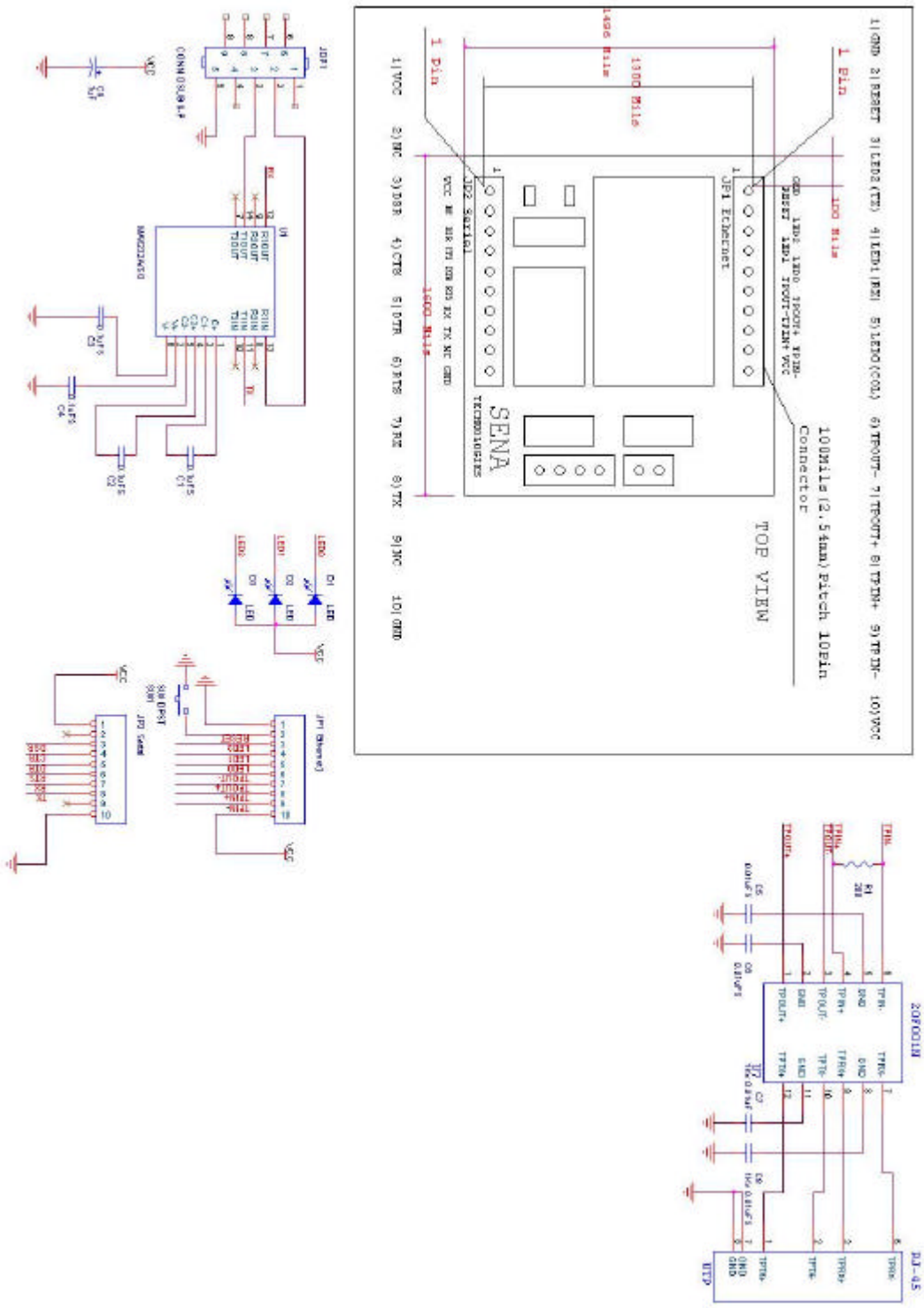


Figure 4.3 HD1321 Connector Pin Assignment and User Device Circuit Diagram

4.2 Installing HelloDevice Utility Software

Run *setup.exe* in the HelloDevice CD-ROM from the PC that is connected to the network. *Setup132x.exe* runs on Microsoft Windows 95/98/ME/NT/2000. The initial screen of *setup132x.exe* is shown below in Figure 4.1

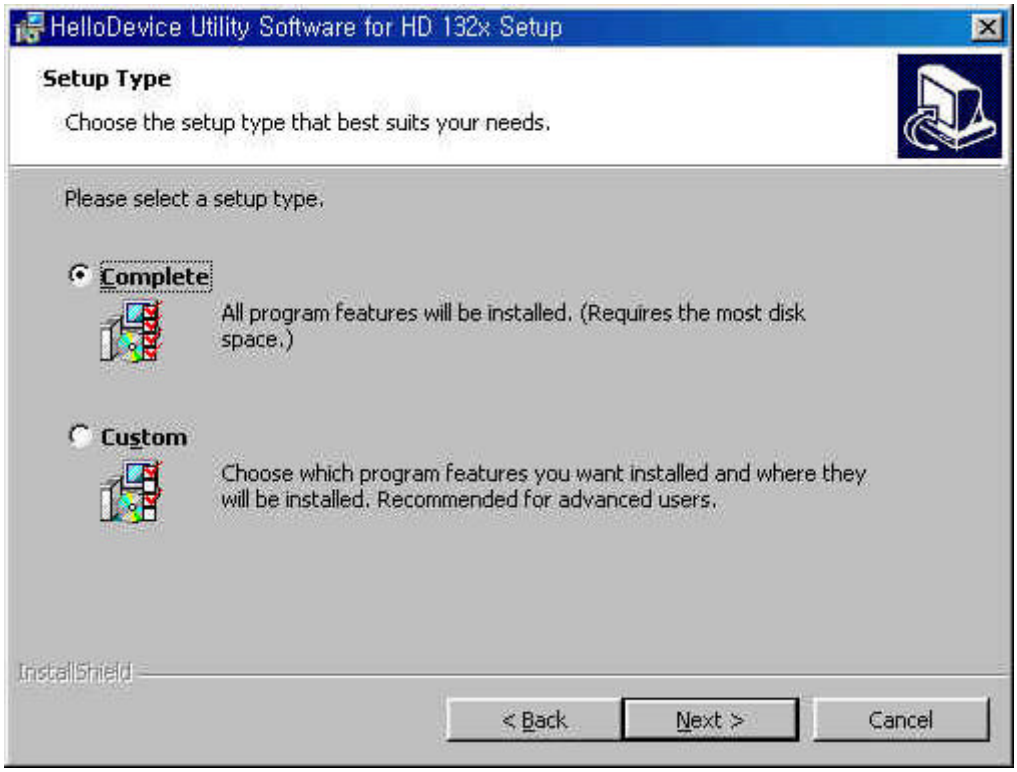


Figure 4.4 Initial Screen of Setup Program

Select **[Complete]** if you want to install the software in a default directory, then the setup program will automatically install the software onto *Program Files**sena**HelloDevice Utility_HD1320*\.

If you want to install the software somewhere else, then select **[Custom]**, and choose a directory folder where the program will be installed. (Figure 4.5)

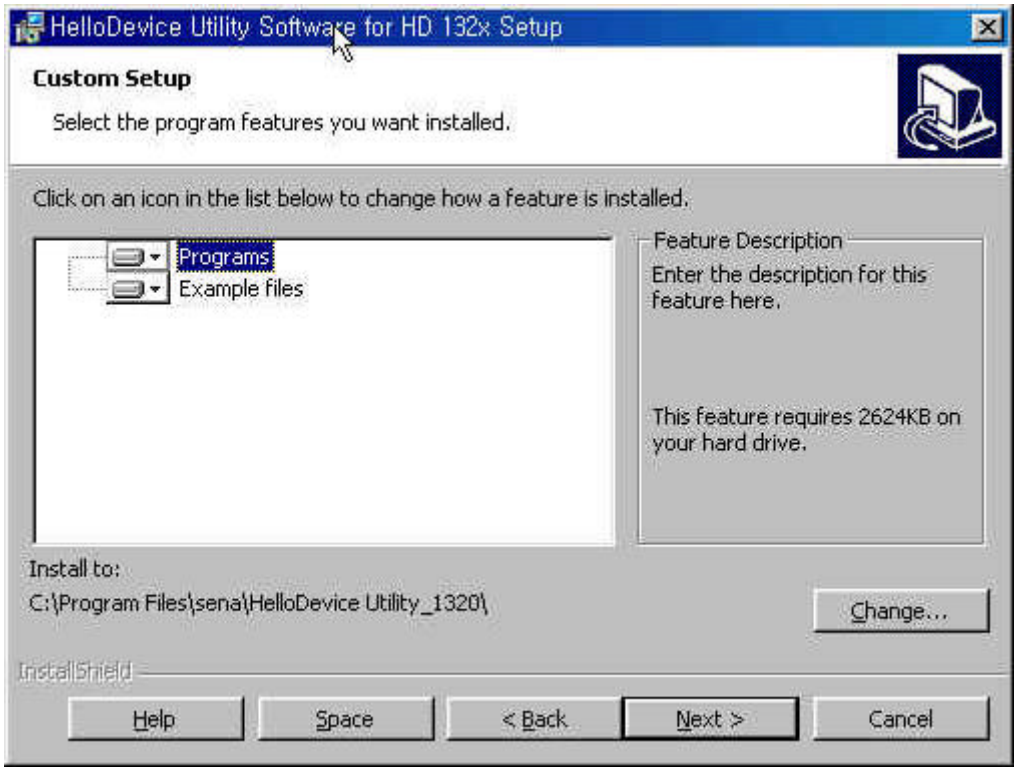


Figure 4.5 Initial Screen of [Custom Setup]

When the installation has been completed, the HelloDevice Utility Software shortcuts are created on your desktop and program menu. Click on the icon and if you see the windows screen as in Figure 4.6, then the installation has been successfully completed.

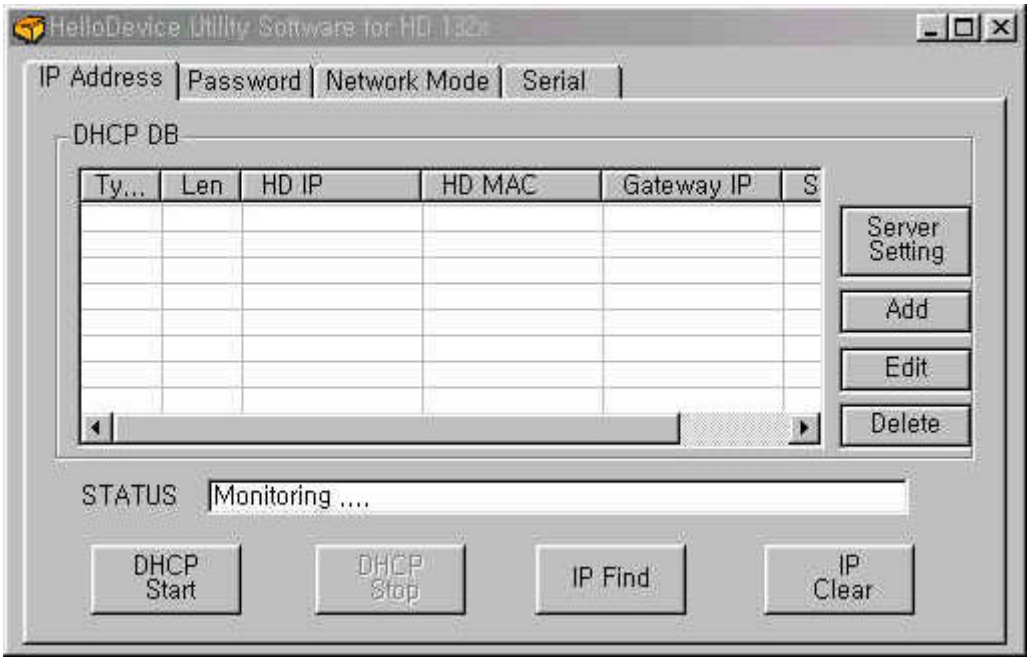


Figure 4.6 Initial Screen of the HelloDevice Utility Software

4.3 Managing an IP address of the HelloDevice

Now we have the HelloDevice physically connected the network and the HelloDevice Utility Software installed on your PC, We will go through the rest of the setup process with the HelloDevice Utility Software in the following order:

1. Assign an IP address of the HelloDevice
2. Set password
3. Set network Mode parameters
4. Set serial parameter

In this section, we will describe how to assign and change an IP address of the HelloDevice, and in the next section, how to set password and parameters.

4.3.1 Assigning an IP Address

Since HD132x Series has the DHCP (Dynamic Host Configuration Protocol) client function, an IP address of the HelloDevice can be assigned with either the HelloDevice Utility Software or any DHCP server in your network. This is based on DHCP (RFC2131) Internet protocol standards. The factory default IP address of HelloDevice is initially set to 0.0.0.0. When the HelloDevice is powered on, the Tx LED of the HelloDevice blinks periodically. That means it requests of an assignment of an IP address.

4.3.1.1 Assigning an IP address from your existing DHCP Server

When you want to assign an IP address of the HelloDevice from your existing DHCP server, the IP address will be automatically assigned from the DHCP server as soon as you turn on the power of the HelloDevice. If Tx LED stops blinking, it means that an IP address has been normally assigned.

4.3.1.2 Assigning an IP address from DHCP Server in the HelloDevice Utility Software

Since the HelloDevice Utility Software supports DHCP server function, it responds to DHCP request of the HelloDevice. The IP address of the HelloDevice is managed by DHCP database in the HelloDevice Utility Software, and you need to enter the MAC[1] address and IP address to DHCP database. MAC

[1] MAC (Media Access Control) address is total 6 bytes with 3 bytes of organization code and 3 bytes of product code, respectively. MAC address of the HelloDevice is composed of 00-01-95, which is a company code, and xx-xx-xx, which is a product code. E.g. 00-01-95-01-aa-08, 00-01-95-01-02-01

address is labeled on NIC (Network Interface Card) of the HelloDevice, and the user specifies an IP address. Consult your network manager for correct network information.

Now, let's assign an IP address with the HelloDevice Utility Software step-by-step.

(1) Launch the HelloDevice Utility Software and select [IP Address] tab.

In **[IP Address]** window, you can set an IP address using DHCP server function (**[DHCP Start]**, **[DHCP Stop]**), and you can trace and clear the IP address (**[IP Find]**, **[IP Clear]**).

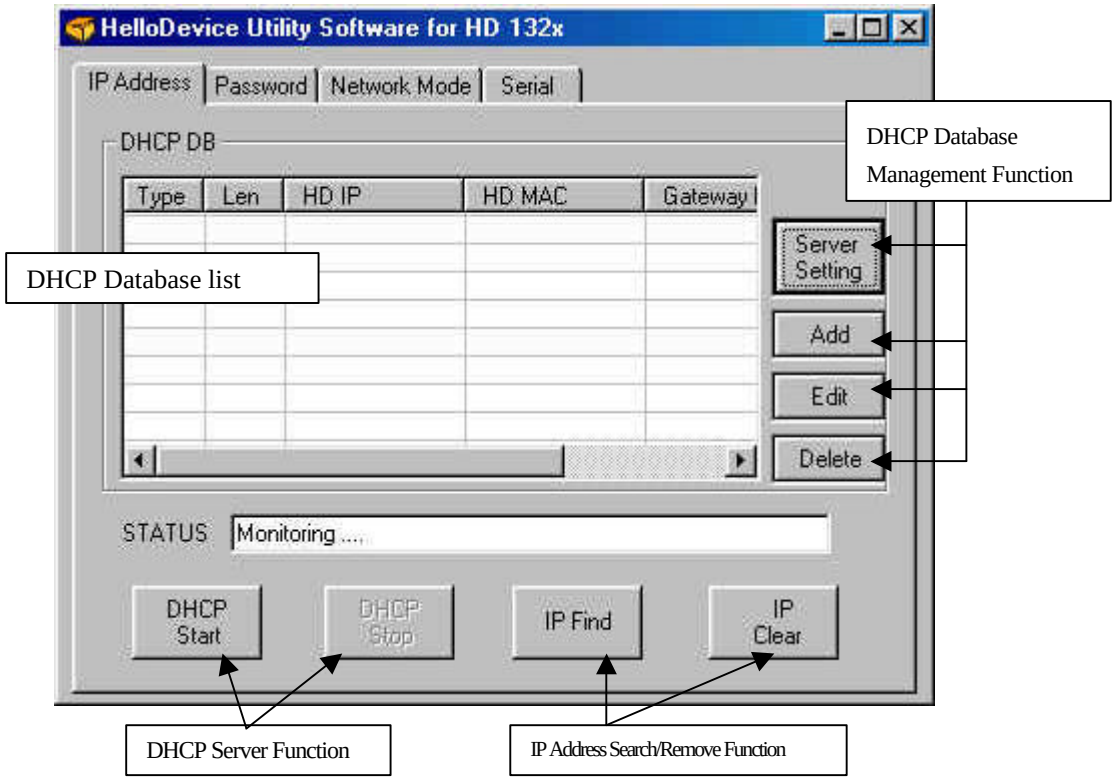


Figure 4.7 [IP Address] Window

(2) Press [Server Setting] button to set parameters of the DHCP server.

In DHCP server setting window (Figure 4.8), you need to enter is [Router] information of the IP address.

[DHCP Server IP] automatically displays the IP address of the user PC where the HelloDevice Utility Software is installed.

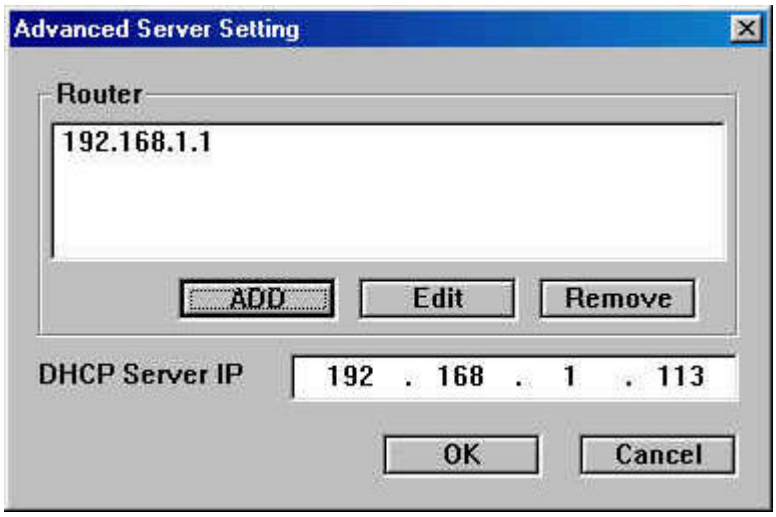


Figure 4.8 [Server Setting] Window

(3) Press [Add] button and enter the DHCP database record.

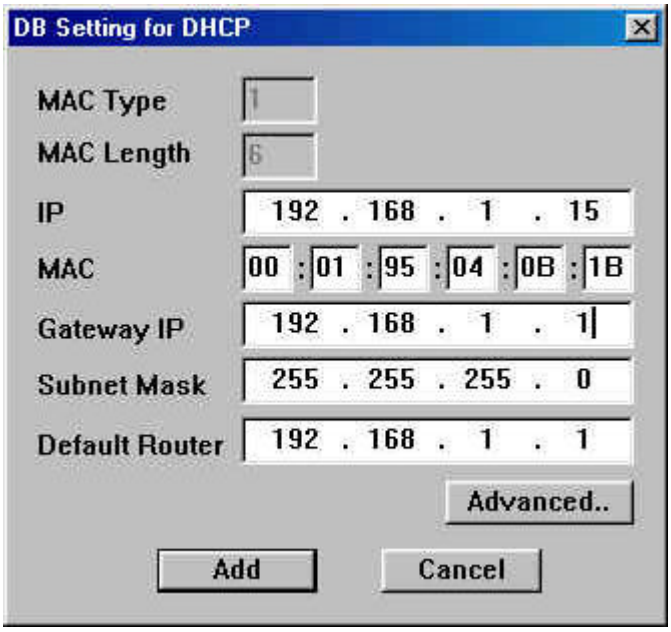


Figure 4.9 [Add] Window

Since the HelloDevice is connected to the Internet through Ethernet, [H/W Address Type] and [H/W Address Length] should always be 1 and 6, respectively.

Enter [Client IP Address] and [Client H/W Address]. [Client IP Address] and [Client H/W Address] are unique IP address and MAC address of the HelloDevice. The MAC address can be found on the sticker labeled on the top of Ethernet IC of the HelloDevice. In Figure 4.9, an example of entry is

shown when the MAC address is 00:01:95:04:0B:1B and the IP address 192.168.1.15.
[Default Router] is an IP address of the router that was set in [Server Setting] window.

(4) Press [Add] button to close the window.

Now, records have been added into the database.

(5) Press [DHCP Start] to start DHCP server function.

Now, the HelloDevice Utility Software operates as a DHCP server. You can see the message in [STATUS] changing from "Monitoring" to "Listening DHCP request". If HelloDevice Utility gets DHCP request from a certain HelloDevice which does not exist, in DHCP DB List, it will run "DB Setting for DHCP" window automatically. User should enter IP address and Gateway address of the HelloDevice, and add them to DHCP DB list.

(6) Check the status of Tx LED of the HelloDevice.

If the Tx LED of the HelloDevice blinks periodically, then it means that the HelloDevice is sending a message requesting for an IP assignment. The HelloDevice Utility Software responds to such request of the HelloDevice. The message in [Status] displays: "DHCP ACK sent... [192.168. 1.15]". And, if the "DHCP ACK sent... [192.168. 1.15]" message is displayed, then it means that an IP address has been assigned for the HelloDevice. In case that the DHCP Server has assigned the IP of HelloDevice, then if the user clicks [DHCP Start Button] in PC having HelloDevice Utility Software which is in same network, the message of "It was Request to other DHCP server! Packet Discarded…" will be displayed.

(7) Confirm the HelloDevice IP address using `ping`.

Execute `ping` at the Command prompt. If set correctly, the following reply will be received:

```
>> ping 192.168.1.15
>> Pinging 192.168.1.15 with 32 bytes of data:
       Reply from 192.168.1.15: bytes=32 time=10ms TTL=251
       Reply from 192.168.1.15: bytes=32 time<10ms TTL=251
       Reply from 192.168.1.15: bytes=32 time=10ms TTL=251
```

If you don't get the message above, repeat steps (5), (6) and (7). Because the IP address has not been assigned correctly.

(8) Confirm the HelloDevice IP address using [IP Find].

When [IP Find] button is pressed, the following window will appear. Enter the MAC address of the HelloDevice and press [Find IP]. The IP address that is currently set will appear in [Found IP] box.

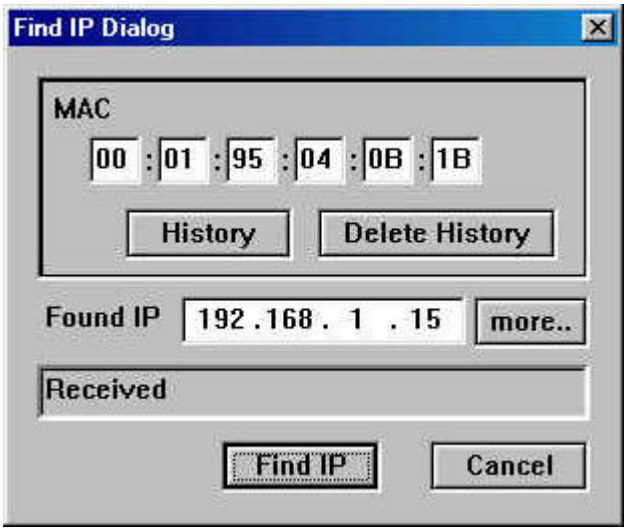


Figure 4.10 [IP Find] Window

If you press [more..], [System info] window will appear, and you can observe information of the current communication status of the HelloDevice. Figure 4.11 shows [System Info] window, and a simple summary of each item in this Figure is shown in Table 4.1.

Note:

1) You can use [Find IP] and other system functions after setting the password parameter. (Refer to Section 4.4.1)
2) You can use [Find IP] only in LAN (Local Area Network) of the same network [Hop]. Collecting information from the remote location over the Internet is impossible.

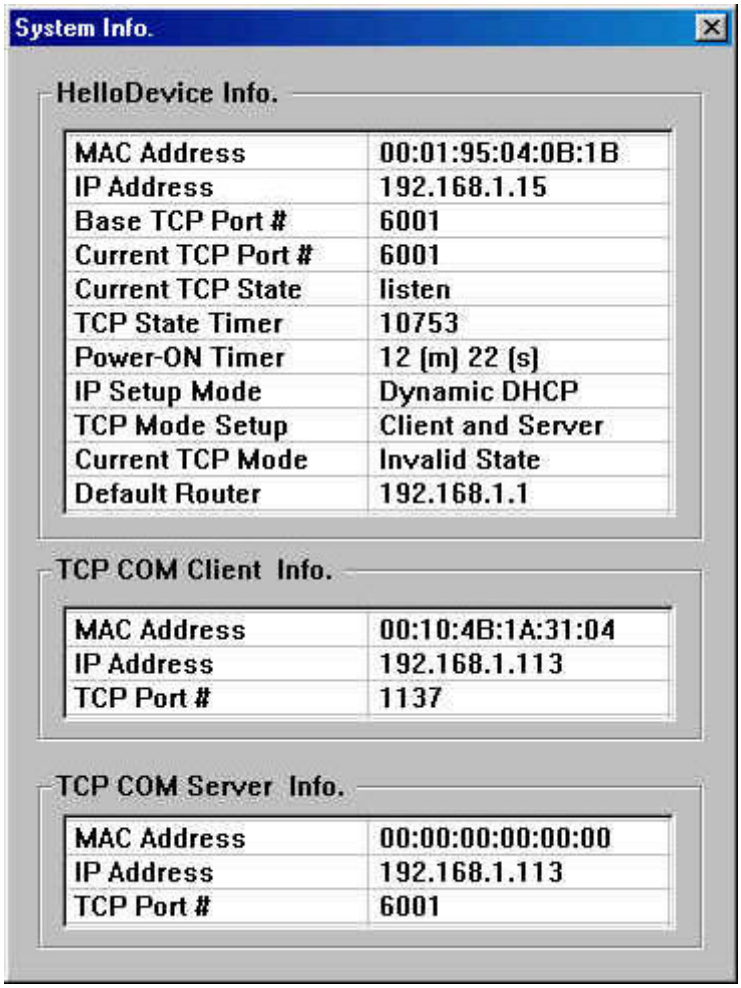


Figure 4.11 [System Info] Window

Group	Symbol	Description	Remark
HelloDevice Info	MAC Address	MAC Address	
	IP Address	IP Address	
	Base TCP port #	Basic TCP port #	
	Current TCP port #	Current TCP port #	
	Current TCP State	Current TCP State	
	TCP State timer	Elapsed time of the TCP State	
	Power-ON timer	Elapsed time after Power-On	
	IP setup mode	IP setup mode	
	TCP mode setup	TCP operation mode setup	
	Current TCP mode	Current TCP operation mode	
	Default Router	IP address of the Default Router	
TCP COM Client info	MAC Address	MAC address of Client	HD is Server or Server/Client mode
	IP Address	IP address of Client	
	TCP port #	Port # of Client	
TCP COM Server info	MAC Address	MAC address of Server	HD is Client or Server/Client Mode
	IP Address	IP address of Server	
	TCP port #	Port # of Server	

Table 4.1. Summary of [System Info]

4.3.2 Changing an IP Address

To change an existing IP address of your HelloDevice, you need to set the IP address to 0.0.0.0 first, and repeat the procedure in Section 4.3.1. In this section, we will explain the procedure assuming we want to change an IP address from 192.168.1.15 to 192.168.1.18.

(1) Get the MAC address

To initialize the existing IP address 192. 168.1.15 to 0.0.0.0, you need to have the MAC address of the HelloDevice with you. The MAC address of the HelloDevice is recorded on the HelloDevice board, or you can also get the address in the DHCP database of the HelloDevice Utility Software.

(2) Initialize the IP address

In [IP Address] window, press [IP Clear], and [Clear IP Dialog] window will pop up. Enter the MAC address of the HelloDevice as shown in Figure 4.12, and press [Clear]. Now the current IP address 192.168.1.15 is initialized to 0.0.0.0.

Note:

If [IP Mode] is set to [Dynamic IP], the IP address cannot be initialized with [IP Clear] command. In this case, the IP address can be initialized when the power is turned off. Refer to Section 4.4.2 for details of [IP Mode].

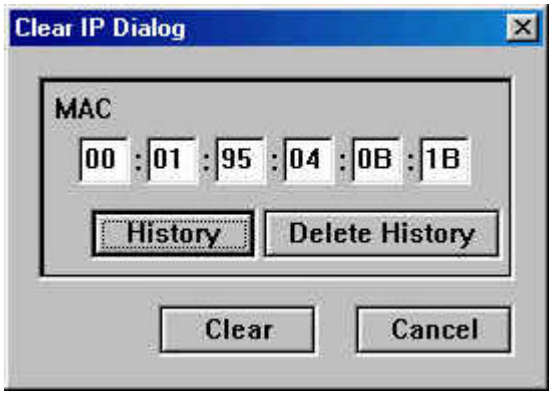


Figure 4.12 IP Address Clear Window

(3) Confirm the initialization of the IP address

Check whether the TX LED of the HelloDevice blinks. If the IP address has been initialized, a message requesting an IP address will be sent through the network, and the TX LED of the HelloDevice will keep blinking.

Note:

If there is a DHCP server in your network, and if [IP mode] is [Dynamic-IP], the DHCP server will immediately assign a new IP address even if the IP address has been initialized. Accordingly, the TX LED will stop blinking. Refer to Section 4.4.2 for details of [IP Mode].

(4) Re-assign the IP address

To re-assign the IP address to 192.168.1.18, edit/modify content of the IP address database with [Edit] command of the DHCP database. Then, repeat the IP address setup procedure in 4.3.1.

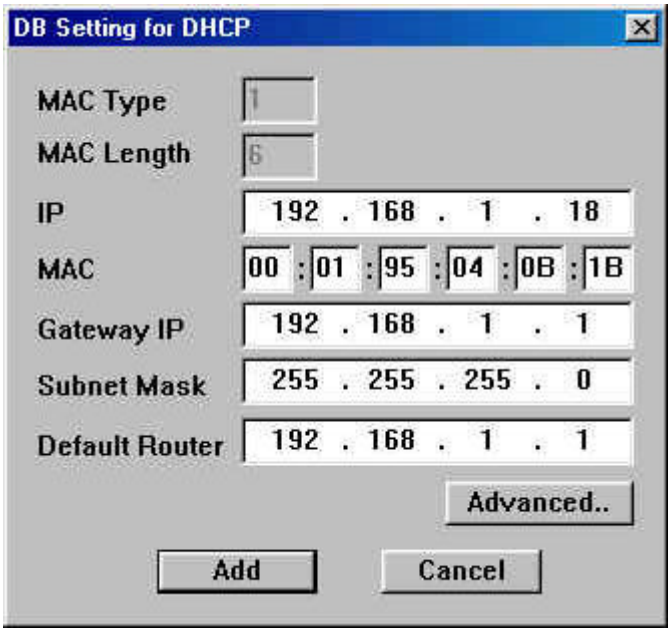


Figure 4.13 [Edit] Window

4.4 Setting Parameters

After the IP address has been assigned successfully to the HelloDevice, you need to set network parameters and serial parameters. To set such parameters, you need to set password parameter first.

Basic setup order of parameters is as follows:

Figure 4.14 Setting the Destination of the Parameter Setup Window

(1) After moving to Parameter window, Enter the target MAC address and IP address of the HelloDevice.

Note:
[MAC address] must be entered, and if [Destination IP] is set to 255.255.255.255, the setup command is transmitted in Broadcast method. Such setup is possible only in LAN environment. You need not have to know the IP address of the HelloDevice in this case.
When setting parameters in HelloDevice located in a remote place (WAN environment), you should enter both the MAC address and the public IP address.

(2) Enter parameters in all fields and press [Send].
By pressing [Send], you can send parameters to the HelloDevice. All parameter settings are sent to the HelloDevice as displayed on the current window.

4.4.1. Password Parameter

For security reason, you must set password parameters in advance before you set other parameters. If you select [Password] Tab, you can verify and change the password.

1) Verifying Password
Default factory setting of the ID is: AAAAAAAA and Password is: AAAAAAAA. After verifying such setting, you can start to set other parameters window. Enter the value in each field as shown in Figure 4.15 and press [Send].

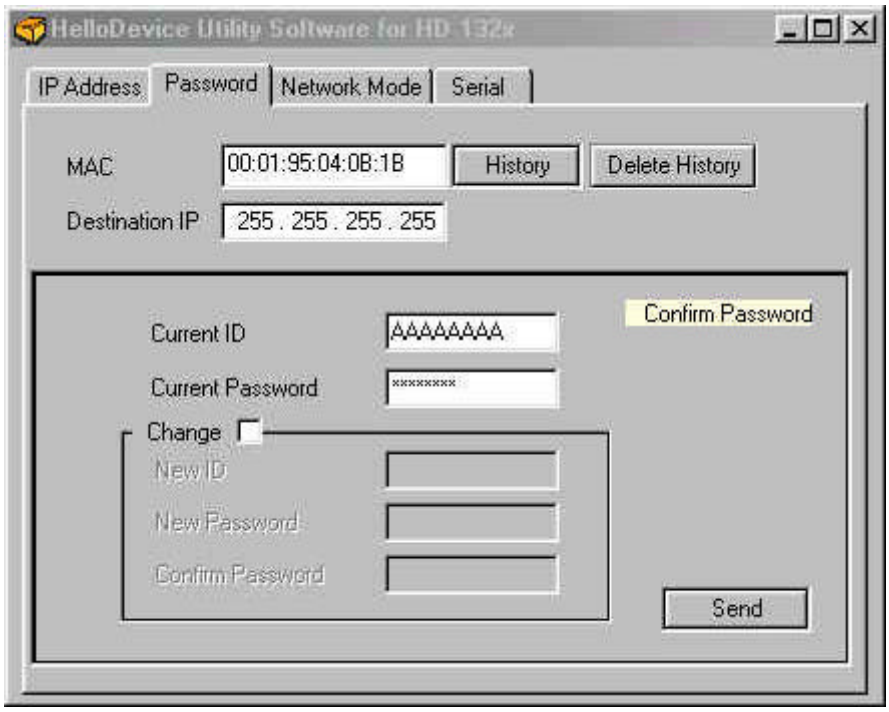


Figure 4.15 Verifying the Password

The password has been successfully confirmed if you see the following window.

Figure 4.16 Confirming Password

2) Changing Password

To set new password, click on [Change] Checkbox, enter the new ID and password, and press [Send]. (See Figure 4.17 and 4.18)

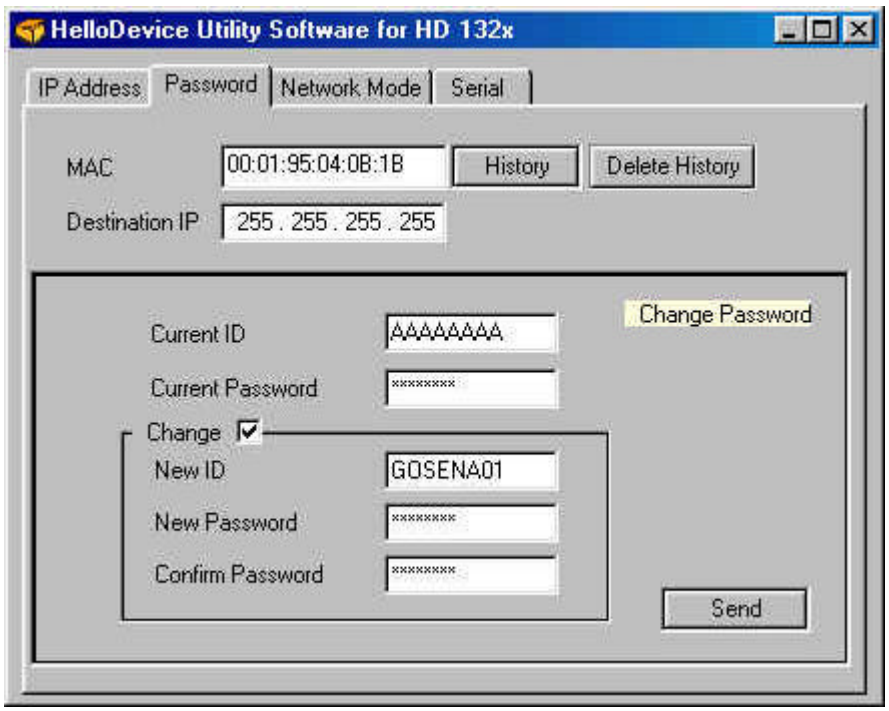


Figure 4.17 Setting New ID & Password

Figure 4.18 Confirming Password Change

ID and Password can be up to 8 characters, and consists of characters, or numbers, or any special symbols. ID and Password are managed by the MAC address, in *PWD.ini* file in the folder where the HelloDevice Utility Software is installed. *PWD.ini* is automatically referred to when you use [IP Clear], [IP Find], and Parameter window.

Note:

If you re-install the HelloDevice Utility Software, the previous PWD file will be deleted also. Therefore, we suggest you always keep the backup file of *PWD.ini*.

4.4.2 System Parameter

In Network Mode window, select setting system parameters. (Figure 4.19)

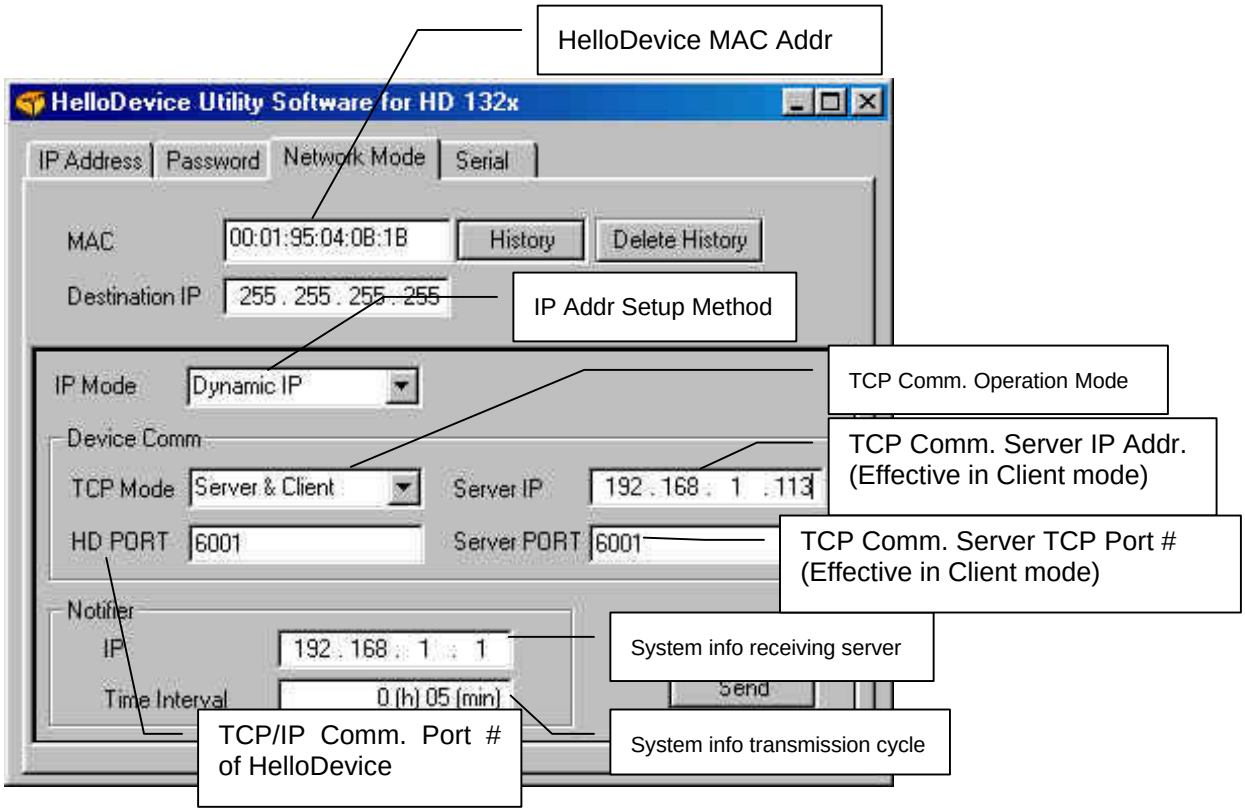


Figure 4.19 Network Mode Window

System parameters are classified as follows:

- Parameters for the HelloDevice
- Parameters for the data server
- Parameters for the remote server that communicates with the HelloDevice

4.4.2.1 Parameters for the HelloDevice

[IP mode], [TCP mode] and [Notifier] are parameters for the HelloDevice. You need to tell what kind of IP the HelloDevice uses, i.e. static or dynamic IP, in [IP mode], and what kind of function it requires, i.e. server, client or server/client, in [TCP mode].

(1)[IP Mode]

Static IP mode

Static IP mode is when your IP address of the HelloDevice is a fixed IP address. After setting the IP address, it will be stored even when the power is turned off. When operating in this mode, the IP address set in 4.3.2 will be in memory unless the IP address is initialized with [IP Clear] command.

Dynamic IP mode

Dynamic IP mode is when your IP address of the HelloDevice is a floating IP address. Under such setting, when the power is restarted, the IP address will be reset through DHCP. DHCP is a method of granting use for an IP address from the DHCP server temporarily. The HelloDevice is programmed to automatically request for an IP reset after 24 hours. Therefore, in the case of dynamic IP mode, the IP address, which was already set, will be maintained for 24 hours.

Note:

In the case of Dynamic IP mode, the IP address cannot be initialized with the [IP Clear] command.

(2) Device Comm

Server mode

This is when the HelloDevice operates as TCP server. The HelloDevice stands by until the TCP client tries to connect to the HelloDevice through the designated port of the HelloDevice. For an initial TCP connection request, the HelloDevice allows connection and transmits the data to the serial device through RS232 port of the HelloDevice. Since the HelloDevice supports only one TCP server socket, the next TCP connection request will be denied if it is currently connected. Other TCP connection requests will be allowed after the current TCP connection is terminated.

Client Mode

This is when the HelloDevice operates as TCP client. When the user device connected to RS232 port of the HelloDevice sends data to the HelloDevice, the HelloDevice tries to connect to a designated remote server through the designated TCP port of the remote server, and send received data to the remote server. However, if the HelloDevice failed to connect to the remote server after 5 attempts at an interval of 1 second, the received data will be deleted.

Server & Client Mode

This is when the HelloDevice operates as TCP server AND client. If you are to transmit data from a serial device to a remote server through the HelloDevice, it will operate as a TCP client. If TCP connection is requested by a remote client, it will operate as TCP server.

Note:

TCP mode can not be changed during RS232 data transmission.

[HD Port]

[HD Port] is the port number of the HelloDevice for TCP connection in TCP Server mode. The range is 2000 ~ 65535.

[Server IP, Server PORT]

When your HelloDevice is set to TCP Client mode, you need to specify the IP address and the port number of a remote server that is to communicate with the HelloDevice. The range [PORT] is 2000 ~ 65535.

(3) [Notifier]

Data server means the server that administers information of the HelloDevice. **This is only when [IP mode] of the HelloDevice is set to Dynamic IP mode AND the environment requires TCP server function (DHCP, TCP server).** since the IP address keeps changing every 24-hour in Dynamic IP mode, there is a need for a data center to administer the IP address and other information of the HelloDevice. To configure a data center, the UDP server socket program which administers setup information that is periodically transmitted from the PC and the HelloDevice, must be running. (Refer to Section 5.3)

The HelloDevice sends its IP address and other information to the data server according at an interval set in [Time Interval]. The range of [Time Interval] is from 90 seconds to 6 hours.

4.4.3 RS232 Communication Parameter

This section describes how to set parameters of the HelloDevice for RS232 communication.

Select [Serial] windows for setting system parameters. (Figure 4.19)

[Baud]

150, 300, 600, 900, 1200, 2000, 2400, 3600, 4800, 7200, 9600, 19200, 38400

[Parity]

One of None, Even, Odd

[Data Bits]

One of 5, 6, 7, 8 bits

[Stopbit]

One of 1, 1 ½ 2 bit

[Handshake]

One of None, H/W, X On/Off

[Time]

[Time] tells the HelloDevice for how long to hold TCP/IP socket when there is no RS232 data input from the user device. Minimum 4 seconds, maximum 100 seconds or Unlimited.

Below is an example of MAC address- 00:01:95:04:0B:1B, 9600 baud rate, Parity None, Data bit 8, Stop bit 1, Handshake None and Time- 50 seconds.

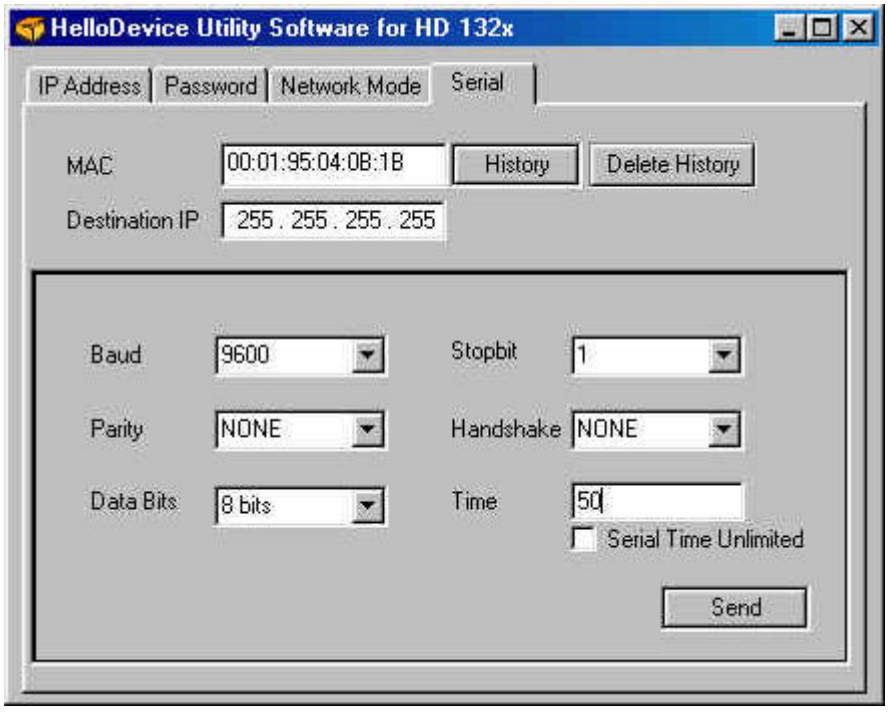


Figure 4.20 Setting Serial Parameters

5. Quick Tour

In this chapter, we will describe how to establish RS232-TCP/IP connection using shareware terminal software such as TeraTerm Pro, HyperTerminal or Telnet, and how to implement such communication using sample socket programs provided in the CD-ROM. And then, we will describe how to implement a data server program using a sample program that manages an IP address and other information of the HelloDevice, when HelloDevice functions as TCP server with a dynamic (floating) IP address. In this chapter, we will use TeraTerm Pro for terminal software to describe each procedure.

5.1 Checking RS232-TCP/IP Communication using TeraTerm Pro

Following items are required to complete the task:

- HD1320E
- RS232 communication cable
- Networking PC
- TeraTerm Pro or terminal program

1) Connect RS232 ports of the PC and the HelloDevice with the RS232 cable. For the cable specification, please refer to Mainboard Specification in Chapter 3.

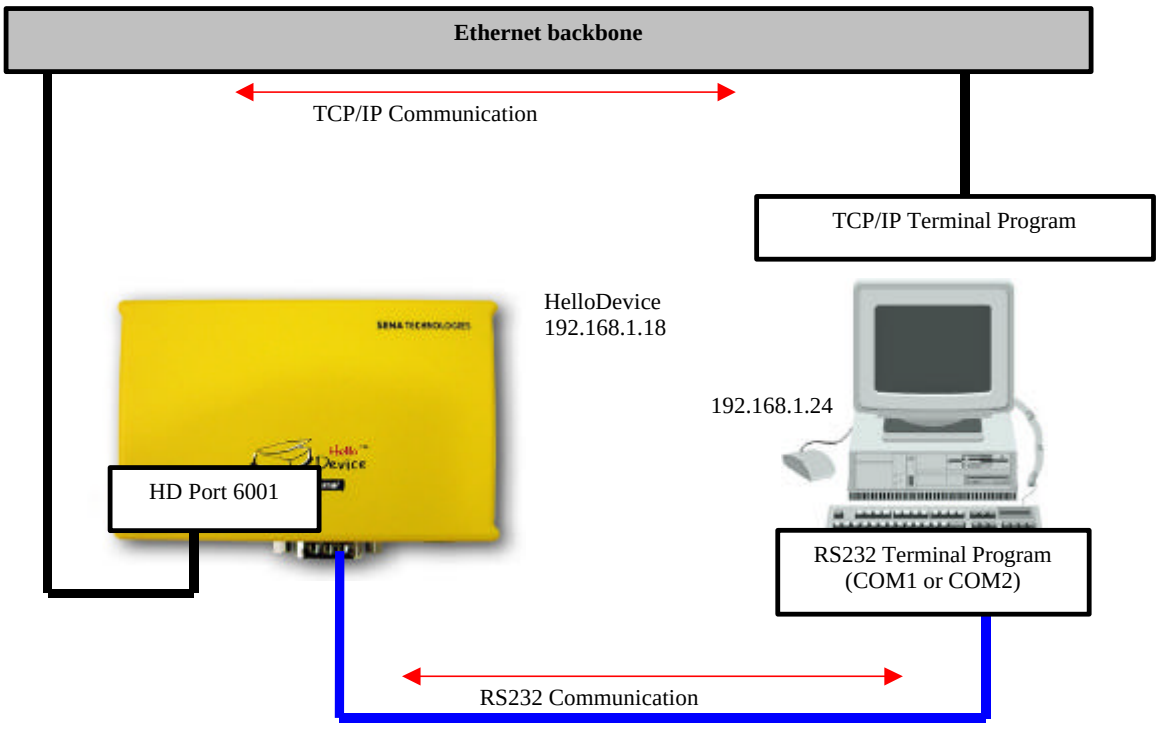


Figure 5.1 Demo Configuration for RS232-TCP/IP Communication

2) Set the TCP mode of the HelloDevice.

TCP mode of the HelloDevice must be set Server, Client or Server/Client, and this is when the HelloDevice functions as TCP Server. In [Network Mode] window, set the TCP mode to **"TCP Server"** and the port to **6001**. (Refer to the TCP topic in 4.4.2) In this section, an IP address of the HelloDevice is assumed to be **192.168.1.18.**

In this TCP Server mode, the HelloDevice functions as TCP server, and allows the connection request from client-side only. Set serial parameters for RS232 communication as follows. For RS232 communication settings, refer to Section 4.4.3.

9600 baud rate
Parity None
Data bit 8
Stop bit 1
Handshake None,
Time 50 seconds

3) Launch TeraTerm Pro for serial communication on the PC.

4) Set the port to COM2 port in TeraTerm Pro.

Parameters of TeraTerm Pro should be the same as serial parameters of the HelloDevice.

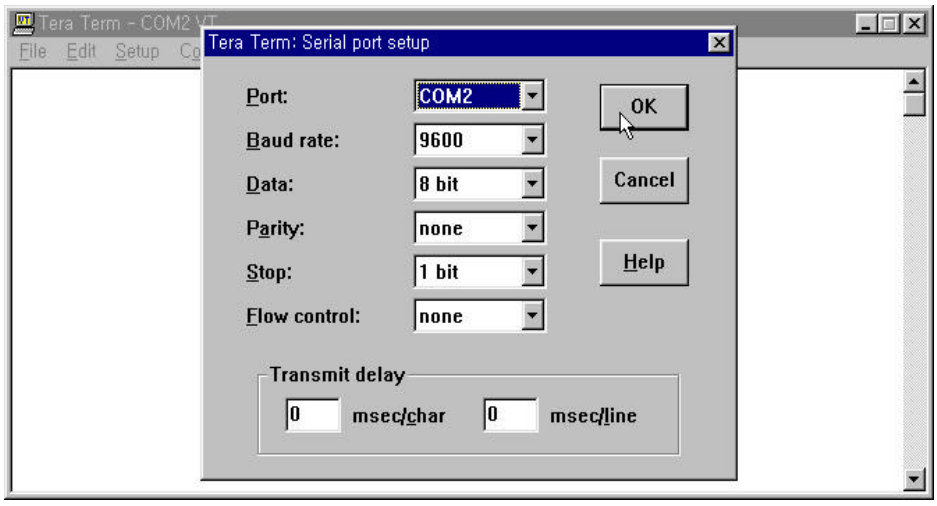


Figure 5.2 Setting RS232 Parameters of TeraTerm Pro

5) Press [OK] button to start the communication emulator.

6) Launch TeraTerm Pro for TCP/IP communication on the PC .

7) Enter the IP address and TCP port number of HelloDevice and press [OK] to request TCP/IP connection to the HelloDevice.

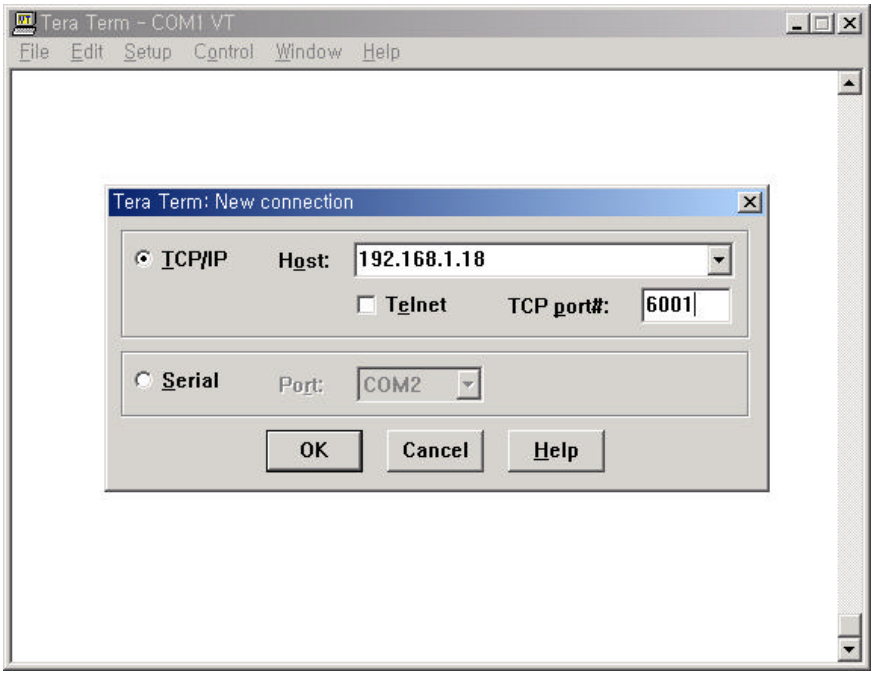


Figure 5.3 TCP/IP Connection on TeraTerm Pro

8) **To check the connection between the PC and the HelloDevice, enter characters on the TCP/IP terminal program and see if these strings appear on the serial terminal program.** Also, enter characters on the serial terminal program and see if these strings appear on the TCP/IP terminal program. Figure 5.4 is a sample screen of such communication.

Note:

[Time] is one of serial parameters of the HelloDevice and can be defined by the user.
If there is no serial communication for a period defined in [Time] parameter, 50 sec in this case, the HelloDevice automatically closes the TCP/IP connection. Therefore, the TCP/IP terminal program window will automatically close if there is no serial communication for 50 sec.

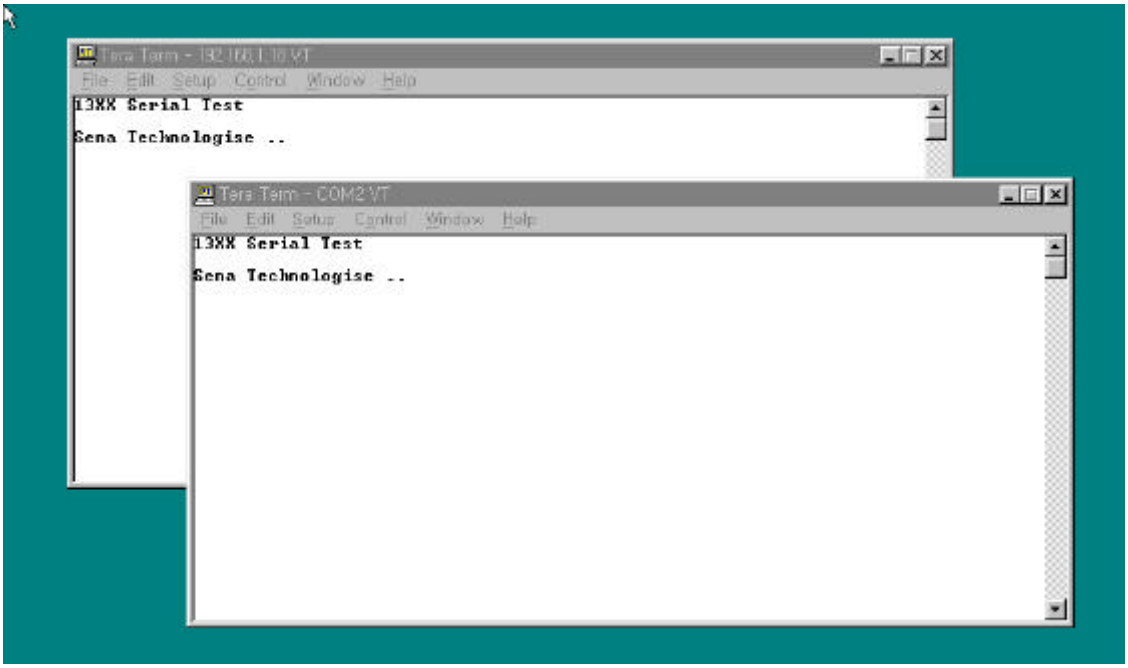


Figure 5.4 RS232-TCP/IP Communication using HelloDevice and TeraTerm Pro

5.2 Checking RS232-TCP/IP Communication using Sample Programs

In this section, we will describe how to establish the RS232-TCP/IP connection using sample programs in the CD-ROM. Figure 5.5 shows the folder where sample programs are located.

- Sample programs when the HelloDevice operates as TCP server (*Server* folder in Figure 5.5)
- Sample programs when the HelloDevice operates as TCP client (*Client* folder in Figure 5.5)
- Sample programs when the HelloDevice sends information to data server periodically (*UDP Server* folder in Figure 5.5)

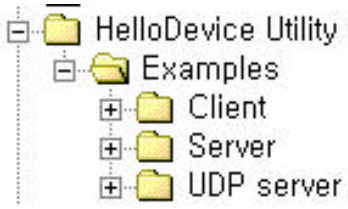


Figure 5.5 Sample Program Folder

Sample programs have been written in Visual C/C++ 6.0 environment, and you can run sample programs in Visual C/C++ 6.0 environment by clicking workspace file (**.dsw*).

5.2.1 When the HelloDevice is TCP Server with a static IP address

The sample program enables the user to connect to the HelloDevice and send strings to the HelloDevice, when the HelloDevice has been set as TCP server mode. The only difference from the previous section is that we use the sample program for TCP/IP terminal program instead of TeraTerm Pro. In this section, we will describe how to establish the connection and check data communication.

1) Connect serial ports of HelloDevice and the PC with the serial cable.

2) Launch serial terminal program such as TeraTerm Pro.

3) Open "*Server.dsw*" file in the "*Server*" folder and "Build" and "Run" the program.

4) Enter the IP address of the HelloDevice.

Figure 5.6 is a sample when the IP address of HelloDevice is 192.168.1.15.

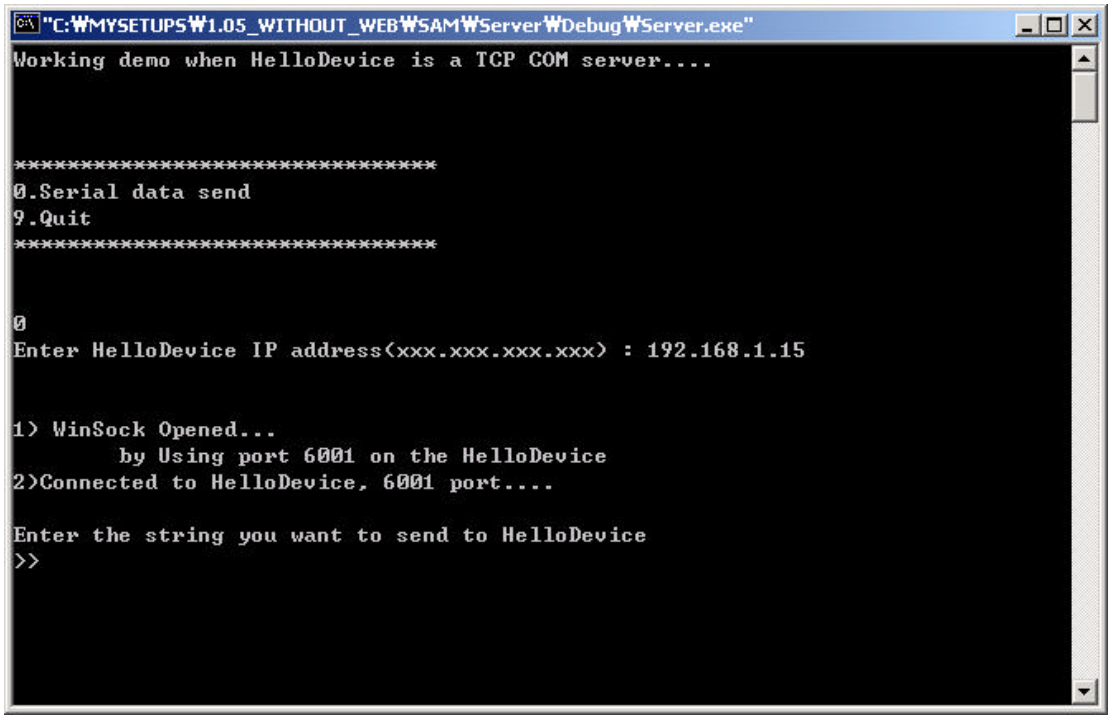


Figure 5.6 Running Server.exe

5) Enter the Character string and press [Enter].

The character string you entered will be displayed on the serial terminal program of the PC.

Below is a major part of the sample program. Main contents are TCP socket initialization, TCP socket open/close and TCP/IP data send functions. For details on window socket programming, refer to

documentations on Microsoft socket programming. Characters in bold font indicate main functions of the socket such as **socket**, **connect** and **send**.

```
//---------------------------------
// Process Serial data send
//---------------------------------
void SerialSend()
{
   char   commandBuf[512]="" ;
   int    commandLen ;
   int    err ;

   // Re-Initialize TCP socket
   TCPSocketInit() ;

   // Read serial data
   //: just ASCII string excluding control characters...
   //: Max size is limited to 512 bytes in this demo....
   printf("\nEnter the string you want to send to HelloDevice\n>>") ;
   scanf("%s", commandBuf) ;

   // Calc serial data length
   commandLen = strlen(commandBuf) ;

   // Send command to HelloDevice
   err = sendto
         (
         sock,
         &commandBuf,
         commandLen,
         0,
         (struct sockaddr*)&serverAddr,
         sizeof(serverAddr)
         ) ;
   if (err == -1 )
   {
      perror("\nsend error\n");
      exit (1);
   }
}

//---------------------------------
// Initialize TCP socket
//---------------------------------
void TCPSocketInit()
{
   char   ipAddrStr[32] ;
   int    clientLen ;
   int    err ;

   // Enter HelloDevice IP address
   printf("Enter HelloDevice IP address(xxx.xxx.xxx.xxx) : ") ;
   scanf("%s", ipAddrStr) ;

   printf("\n\n") ;

   // Convert IP address from string to long
```

```
    ipAddr = decodeAddress(ipAddrStr) ;

    // Windows requires that winsock be initialized.
    err = WSAStartup (0x0101, &lpWSAData);
    if ( err != 0 )
    {
       printf("\nCannot open WinSock???\n");
       exit (1) ;
    }
    else
       printf("1) WinSock Opened...\n") ;

    // Create TCP socket
    clientLen = sizeof(serverAddr);
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if ( sock < 0 )
    {
       perror("\nsocket error???\n");
       exit (1) ;
    }

    // Clear IP address fields
    memset( (char*) &addr, 0, sizeof( addr ) );
    memset( (char*) &serverAddr, 0, sizeof( serverAddr ) );

    // Set my IP address : TCP port 6001
    addr.sin_family = AF_INET;
    // You may use any port other than 6001 in host side!
    addr.sin_port = htons(6001);
    addr.sin_addr.s_addr = INADDR_ANY;

    // Set HelloDevice IP address : TCP port 6001
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(6001);
    printf("\tby Using port %d on the HelloDevice\n",
              ntohs(serverAddr.sin_port) );
    serverAddr.sin_addr.s_addr = htonl(ipAddr);

    // Connecting to HelloDevice
    err = connect( sock, (struct sockaddr*) &serverAddr,
                    sizeof(serverAddr) );
    if ( err == -1 )
    {
       perror("\nCannot connect to HelloDevice???\n");
       exit (1) ;
    }
    printf("2)Connected to HelloDevice, %d port....\n",
          ntohs(serverAddr.sin_port) );
}

//------------------------------
// Close TCP socket
//------------------------------
void TCPSocketClose()
{
    closesocket(sock) ;
}
```

5.2.2 When the HelloDevice is TCP Client

This is the case when the HelloDevice is TCP Client and the sample program on the PC TCP Server. The scenario of the sample program is as follows:

1. The serial device sends data to the HelloDevice through serial ports.
2. The HelloDevice receives data from the serial device.
3. The HelloDevice connects to the PC over the Internet. The IP address of the PC is stored in the HelloDevice.
4. The HelloDevice sends data to the PC over the Internet.
5. The PC receives data from the HelloDevice over the Internet.
6. The PC resends data to the HelloDevice over the Internet.

1) **Connect serial ports of HelloDevice and the PC with the serial cable.**

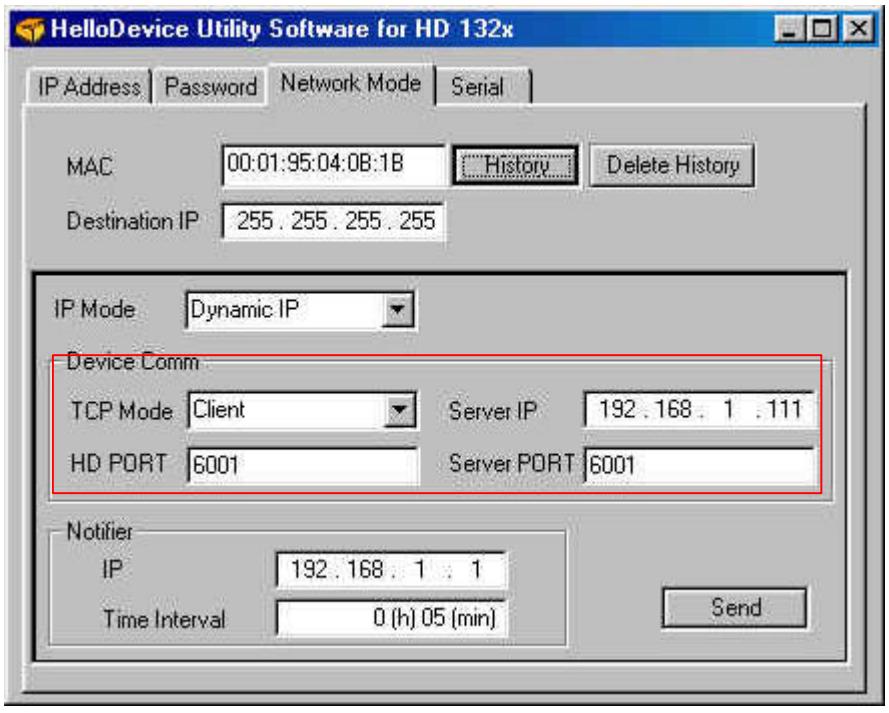


Figure 5.7 Setting [TCP Mode] of the HelloDevice to "Client"

2) **Set System parameters**

- Set [TCP mode] to "Client"
- Set [Server IP] to an IP address of the PC where the sample server program is running
- Set [Server PORT] to 6001.
- Other parameters remain the same as the sample in 5.2.1. Figure 5.7 is an example setup when an IP address of the PC is 192.168.1.111.
- Press [Send].

Now the HelloDevice is set to function as TCP Client and to send data to a designated data server.

3) **Set [Time] to "Serial Time Unlimited" in serial configuration of the HelloDevice and press [Send] as shown in Figure 5.8.**

The HelloDevice is set to keep TCP connection alive as long as the sever program does not close the connection.

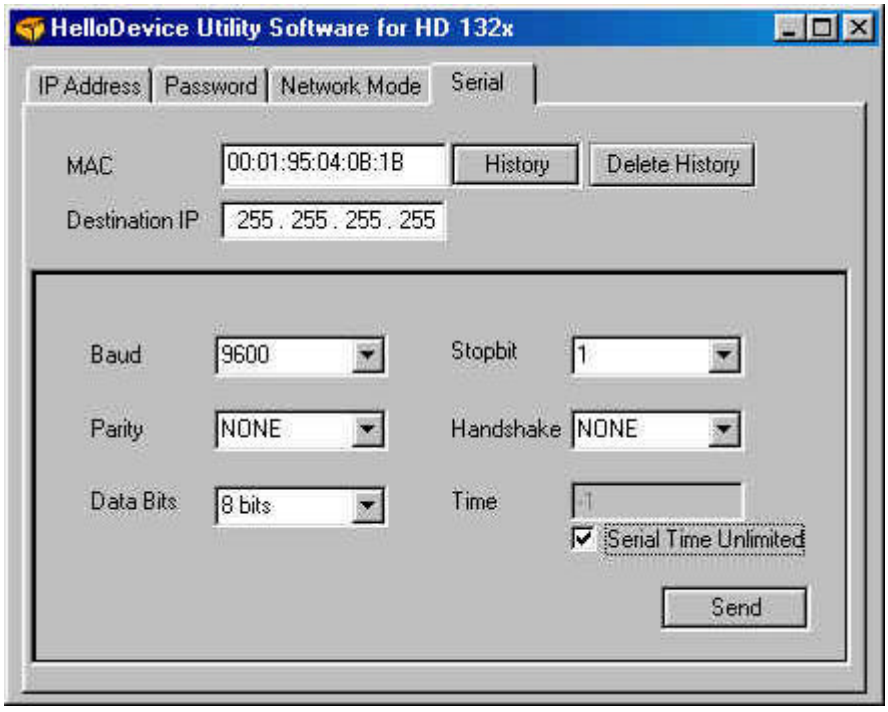


Figure 5.8 Setting [Time] to "Serial Time Unlimited"

4) Launch serial terminal program such as TeraTerm Pro.

5) Open "*Client.dsw*" file in "Client" folder and "Build" and "Run" the program.

6) While the sample program is running, send character strings from the serial terminal program.

Character strings that you entered will be displayed as received data on the sample program, and, since the sample program is programmed to send received data back to the HelloDevice, the serial terminal program will again display the character strings.

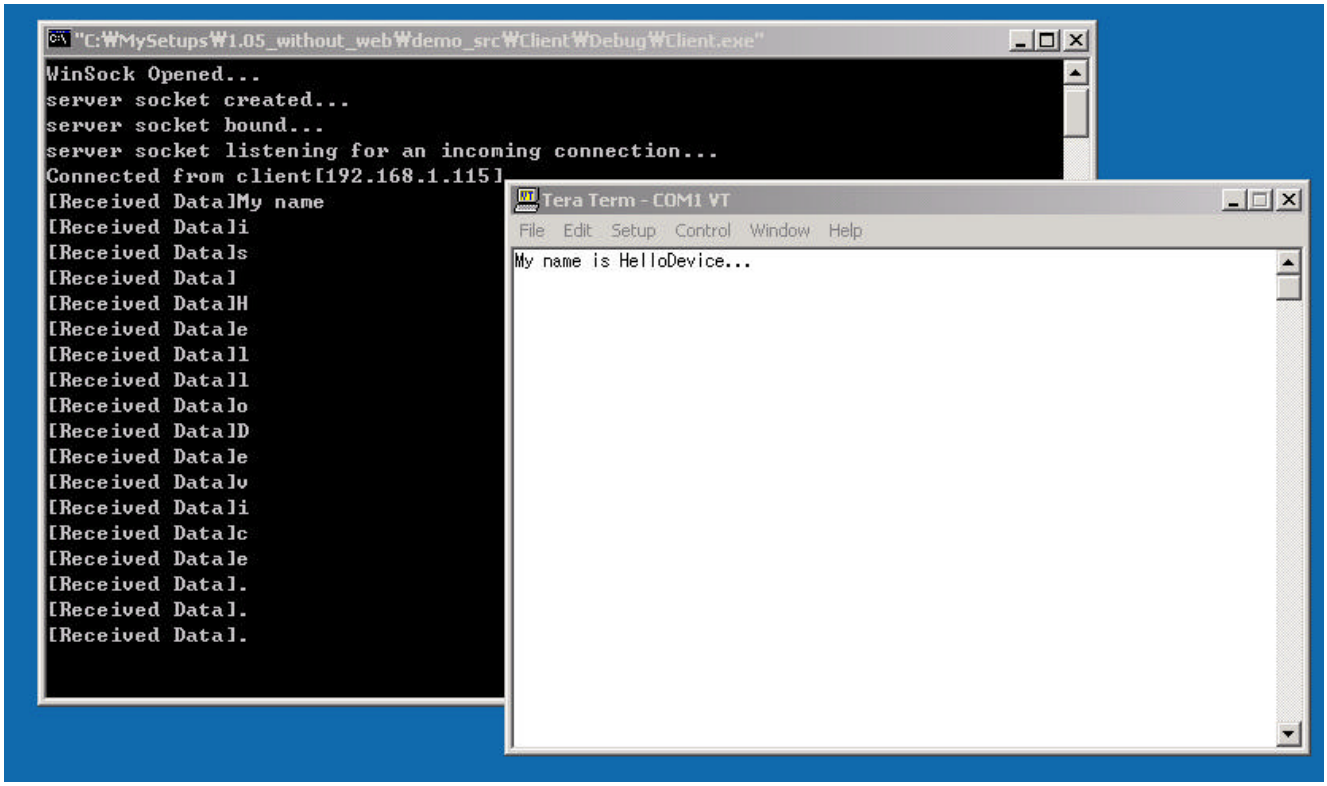


Figure 5.9 Running the Sample Program

Below is a major part of the sample program. Characters in bold font indicate main functions of the server socket such as **listen**, **bind**, **accept**, **socket**, **recv** and **sendto**. Since this sample is related to the server socket, keep in mind that this is different from the sample of 5.2.1 such as socket generation and data transmission.

```
//----------------------------------------
// Process listening : ProcessRecv called
//----------------------------------------
void ProcessListen()
{
    int  clientLen;
    char chDumm;
    clientLen = sizeof(addrClient);
    memset((char*) &addrClient, 0, sizeof(addrClient));
```

```
   while (1) {
   //------------------------------------------------------------
   // accept an incoming connection attempt on the server socket
   //------------------------------------------------------------
      sockClient = accept(sock,(LPSOCKADDR)&addrClient,&clientLen);
      if (sockClient == INVALID_SOCKET) {
         printf("\naccept error???\n");
         printf("\nPlease, try later(press ENTER)");
         scanf("%c", &chDumm);
         break;
      } else {
         printf("Connected from client[%d.%d.%d.%d]\n"
            ,addrClient.sin_addr.S_un.S_un_b.s_b1
            ,addrClient.sin_addr.S_un.S_un_b.s_b2
            ,addrClient.sin_addr.S_un.S_un_b.s_b3
            ,addrClient.sin_addr.S_un.S_un_b.s_b4);
   //------------------------------------------------------------
   // receives data from the client socket(HD1320)
   //------------------------------------------------------------
         if(ProcessRecv()) { // Exit Program
            break;
         } else {          // Continue Listen
                  // Just close the client socket
            TCPSocketClose(CLOSE_CLIENT_SOCKET);
            memset((char*) &addrClient, 0, sizeof(addrClient));
         }
      }
   }
}

//---------------------------------------------
// Process receive : socket recv function called
//  return  9 : exit program
//         0 : continue listening
//---------------------------------------------
int ProcessRecv()
```

```
{
   char bufReceived[BUFSIZE];
   int  nCountReceived = 0;
   int  nCommand = 0;
   int  nShowMenu = 0;
   memset(bufReceived,'\0',sizeof(bufReceived));
   while(1) {
//-----------------------------------------------------------
//  receives data from a connected socket(HD1320)
//-----------------------------------------------------------
       nCountReceived = recv(sockClient,bufReceived,sizeof(bufReceived),0);
       if (nCountReceived == 0) { // HelloDevice closed the client socket
          printf("Disconnected from client[%d.%d.%d.%d]\n"
              ,addrClient.sin_addr.S_un.S_un_b.s_b1
              ,addrClient.sin_addr.S_un.S_un_b.s_b2
              ,addrClient.sin_addr.S_un.S_un_b.s_b3
              ,addrClient.sin_addr.S_un.S_un_b.s_b4);
          DisplayMenu();
          scanf("%d", &nCommand);
          return nCommand;
       } else if (nCountReceived < 0) {
          if (++nShowMenu == SHOWMENU) {
              DisplayMenu();
              scanf("%d", &nCommand);
              return nCommand;
          } else {
              continue;
          }
       } else { // received some data
          // Send echo to client
       if(!SerialSend(bufReceived, nCountReceived)){ // Sending echo error
              printf("Fail to send echo to client[%d.%d.%d.%d]\n"
                  ,addrClient.sin_addr.S_un.S_un_b.s_b1
                  ,addrClient.sin_addr.S_un.S_un_b.s_b2
                  ,addrClient.sin_addr.S_un.S_un_b.s_b3
                  ,addrClient.sin_addr.S_un.S_un_b.s_b4);
              DisplayMenu();
```

```
            scanf("%d", &nCommand);
            return nCommand;
         }
         printf("[Received Data]%s\n", bufReceived);
         nCountReceived = 0;
         nShowMenu = 0;
         memset(bufReceived,'\0',sizeof(bufReceived));
      }
   }
}

//----------------------------------------
// Process sending serial data to HelloDevice
//     return : 1 - success , 0 - failure
//----------------------------------------
int SerialSend(char* strReceived, int nReceived)
{
   int err;
   char* pCommandBuf = (char*)malloc(nReceived + 2);
   memcpy(&pCommandBuf[0], strReceived, nReceived);
   // Send command to HelloDevice
   err = sendto
         (
         sockClient,
         pCommandBuf,
         nReceived + 2,
         0,
         (struct sockaddr*)&addrClient,
         sizeof(addrClient)
         ) ;
   free(pCommandBuf);
   if (err == -1 )
   {
      return 0;
   }
   return 1;
}
```

```
//---------------------------------
// Initialize TCP server socket
// return : 1 - success , 0 - failure
//---------------------------------
int TCPServerSocketInit()
{
   char   chDummy;
   int    err ;

   //-----------------------------------------------------
   // Initiate use of WS2_32.DLL by a process
   //-----------------------------------------------------
   err = WSAStartup (0x0101, &lpWSAData);
   if ( err != 0 )
   {
      printf("\nfail to start up winsock???\n");
      scanf("\nPlease, try later(press ENTER)");
      scanf("%c", &chDummy);
      return 0;
   }
   else
      printf("WinSock Opened...\n") ;

   //-----------------------------------------------------
   // create a server socket
   //-----------------------------------------------------
   sock = socket(AF_INET, SOCK_STREAM, 0);
   if ( sock < 0 )
   {
      printf("\nsocket error???\n");
      printf("\nPlease, try later(press ENTER)");
      scanf("%c", &chDummy);
      return 0;
   }
   printf("server socket created...\n") ;
```

```
    // Clear server IP address fields
    memset( (char*) &addr, 0, sizeof( addr ) );
    // Set server IP address : TCP port 6001
    addr.sin_family = AF_INET;
       // You may use any port other than 6001 in host side!
    addr.sin_port = htons(6001);
    addr.sin_addr.s_addr = INADDR_ANY;
    //-----------------------------------------------------
    //  associate a local address with a socket
    //-----------------------------------------------------
    if(bind(sock,(LPSOCKADDR)&addr,sizeof(addr))
        == SOCKET_ERROR) {
        printf("\nserver socket bind error???\n");
        scanf("\nPlease, try later(press ENTER)");
        scanf("%c", &chDummy);
        return 0;
    }
    printf("server socket bound...\n") ;
    //-----------------------------------------------------
    //  places a socket a state where it is listening for an
    //      incoming connection.
    //-----------------------------------------------------
    if(listen(sock,5) == SOCKET_ERROR) {
        printf("\nserver socket listen error???\n");
        scanf("\nPlease, try later(press ENTER)");
        scanf("%c", &chDummy);
        return 0;
    }
    printf("server socket listening for an incoming connection...\n");
    return 1;
}

//-------------------------------------------------------------------
// Close TCP socket
// parameter
//bCloseOnlyClient : CLOSE_CLIENT_SOCKET(1) - close just client socket
//CLOSE_ALL_SOCKET(0) - close all socket and clean up
```

```
//-------------------------------------------------------------------
void TCPSocketClose(int bCloseOnlyClient)
{
   //-----------------------------------------------------
   // close the client socket
   //-----------------------------------------------------
   if (sockClient != INVALID_SOCKET) {
      closesocket(sockClient);
      sockClient = INVALID_SOCKET;
   }

   if (!bCloseOnlyClient) {
      //-----------------------------------------------------
      // close the server socket
      //-----------------------------------------------------
      closesocket(sock);
      //-----------------------------------------------------
      // terminate use of the WS2_32.DLL
      //-----------------------------------------------------
      WSACleanup();
   }
}
```

5.2.3 When the HelloDevice is TCP server with a dynamic IP address

This is the case when you want to use the HelloDevice as TCP server in DHCP environment. In DHCP environment, the HelloDevice has a dynamic IP address, and therefore, the user can't know the IP address of the HelloDevice because the IP address of the HelloDevice keeps changing.
To solve this problem, the sample program enables users to track the IP information of the HelloDevice. The HelloDevice includes a function that periodically sends IP information of the HelloDevice to a data server designated by the user. Therefore, to use the HelloDevice as TCP server with a dynamic IP address, you should connect to the data server first to obtain the IP address, and then you can connect to the HelloDevice with the obtained IP address.

In case of WAN, for the HelloDevice to send information, the data server needs to have a public static IP address. In case of LAN, for the HelloDevice to send information, the data server needs either public static IP address or private static IP Address.

The sample program for this function has been implemented in UDP (User Datagram Protocol). When the HelloDevice sends its MAC address, local port and IP address to the IP address of the data server through UDP 514 port, the sample program receives and displays the information. For setup of data server parameters, refer to Section 4.4.2.
The HelloDevice transmits data that consist of a total of 14 bytes, and the form is as follows:

“OK” (2 Byte)+ MAC address(6 Byte) + Local port(2 Byte) + IP address(4 Byte)

Below is a source sample of the data server.

```
/*************************************************************************
    UDP-based data server sample program

    HelloDevice notifying message format

    : total 14 bytes data
    : The number in the parenthesis means the byte size...

    'O'(1)+'K'(1)+ MAC address(6) + local port number(2) + IP address(4)

      e.g.
        4f-4b-00-01-95-04-04-01-17-71-c0-a1-a8-0f
*************************************************************************/
#include <stdio.h>
#include <time.h>
#include <winsock.h>


// Global Variable definition
WSADATA lpWSAData;      // Socket data structure
SOCKADDR_IN  addrFrom;
int    sock ;        // Socket
int byte_received ;    // byte received
struct sockaddr_in addr ;     // My IP address
```

```
// UDP Socket function
void UDPSocketCreate() ;
void UDPSocketRun() ;
void UDPSocketClose() ;

// Main function
void main()
{
   printf("UDP Hello Device Program \n");

   // 1) Socket Creation
   UDPSocketCreate() ;

   while(1)
   {
      // 2) Listen until any incoming data
      // 3) Receive if any incoming data
      // 4) Print data
      UDPSocketRun();
   }
   // 5) Close UDP Socket
   UDPSocketClose() ;
}

//---------------------
// UDP Socket Creation
//---------------------
void UDPSocketCreate()
{

   // Windows requires that winsock be initialized.
   if (WSAStartup (0x0101, &lpWSAData) == INVALID_SOCKET)
   {
      printf("\nCannot open WinSock???\n");
      exit (1) ;
   }
   else
```

```
        printf("WinSock Opened...waiting..\n") ;

    // Create Windows socket for UDP
    sock = socket(AF_INET, SOCK_DGRAM,0);
    if (sock < 0)
    {
        perror("\nsocket error???\n");
        exit (1) ;
    }

    // Setupo the port configuration
    // UDP port : 514
    addr.sin_family = AF_INET;
    addr.sin_port = htons(514);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    // Launch UDP socket
    if (bind(sock,(LPSOCKADDR)&addr,sizeof(addr)) == SOCKET_ERROR)
    {
        printf("\n Socket error program terminated..\n " );
        exit(1);
    }
}
//----------------------
// UDP Socket Run
//----------------------
void UDPSocketRun()
{
    int nAddrFromLen = sizeof(addrFrom);
    IN_ADDR inFrom;
    char test_buff[300];
    char dbuffer [9];
    char tbuffer [9];
    int i ;

    // Wait until it receives data
    byte_received =
```

```
recvfrom(sock,test_buff,250,0,(LPSOCKADDR)&addrFrom,&nAddrFromLen);
   if (byte_received==SOCKET_ERROR)
   {
      printf("\n Socket error program terminated..\n");
      exit(1);
   }
   memcpy(&inFrom, &addrFrom.sin_addr,4);

   // Calculate date & time
   _strdate( dbuffer );
   printf( "\n Info. was notified on %s ", dbuffer );
   _strtime( tbuffer );
   printf( "%s \n", tbuffer );

   // Decode the incoming datagram
   printf("\n from %s \n\n MAC address:  " , inet_ntoa(inFrom) );

   for(i=2; i<8; i++)
      printf("%2x " ,(unsigned char)test_buff[i]);
   printf("\n port # :");

   for(i=8;i<10;i++)
      printf("%3x",(unsigned char)test_buff[i]);

   printf(" \n IP address :");

   for(i=10;i<14;i++)
      printf(" %3d ",(unsigned char)test_buff[i]);

   printf("\n \n \n \n");

}
```

```
//---------------------
// UDP Socket Close
//---------------------
void UDPSocketClose()
{
   closesocket(sock);
}
```

The sample program has been written in Visual C/C++ 6.0 environment, and you can run it in Visual C/C++ 6.0 environment by clicking workspace file (*.*dsw*). Figure 5.10 shows data received from the HelloDevice.

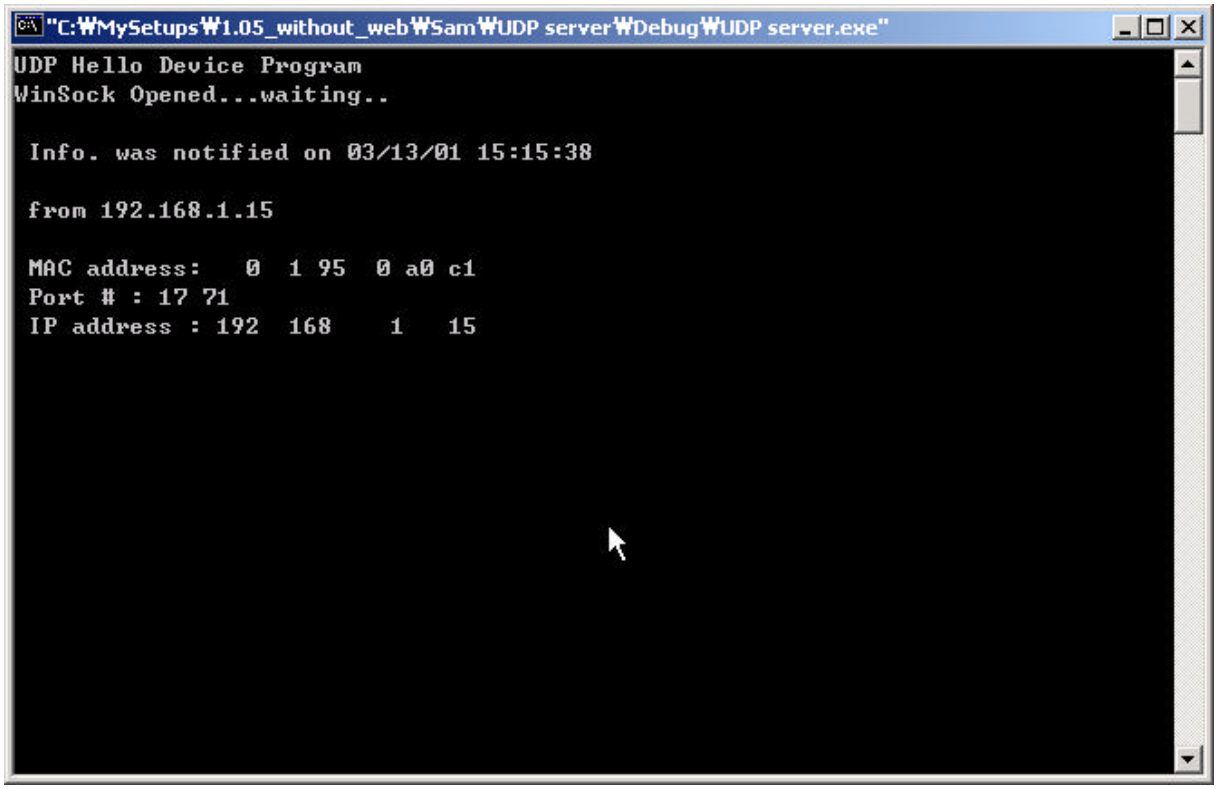


Figure 5.10 Running Data Server Sample Program

Appendix. RS232 Cable Connection

Signals of general RS232 9 pin connector:

Pin 1 Received Line Signal Detector (Data Carrier Detect)	DCD
Pin 2 Received Data	RxD
Pin 3 Transmit Data	TxD
Pin 4 Data Terminal Ready	DTR
Pin 5 Signal Ground	GND
Pin 6 Data Set Ready	DSR
Pin 7 Request To Send	RTS
Pin 8 Clear To Send	CTS
Pin 9 Ring Indicator	RI

Table A. Signals of general RS232 9 Pin Connector

HD1320 and 1320E support RxD, TxD, RTS and CTS pins. If DTR and DSR are required for the flow control, you can disconnect pins 1(DCD), 4(DTR), 6(DSR) of the RS232 connector as shown in Figure 6.1 to allow flow control communication with only RTS and CTS.

In case of HD1321, 5V TTL signal is provided for RS232 communication, and it supports RXD, TXD, RTS, CTS, DTR and DRS signals.

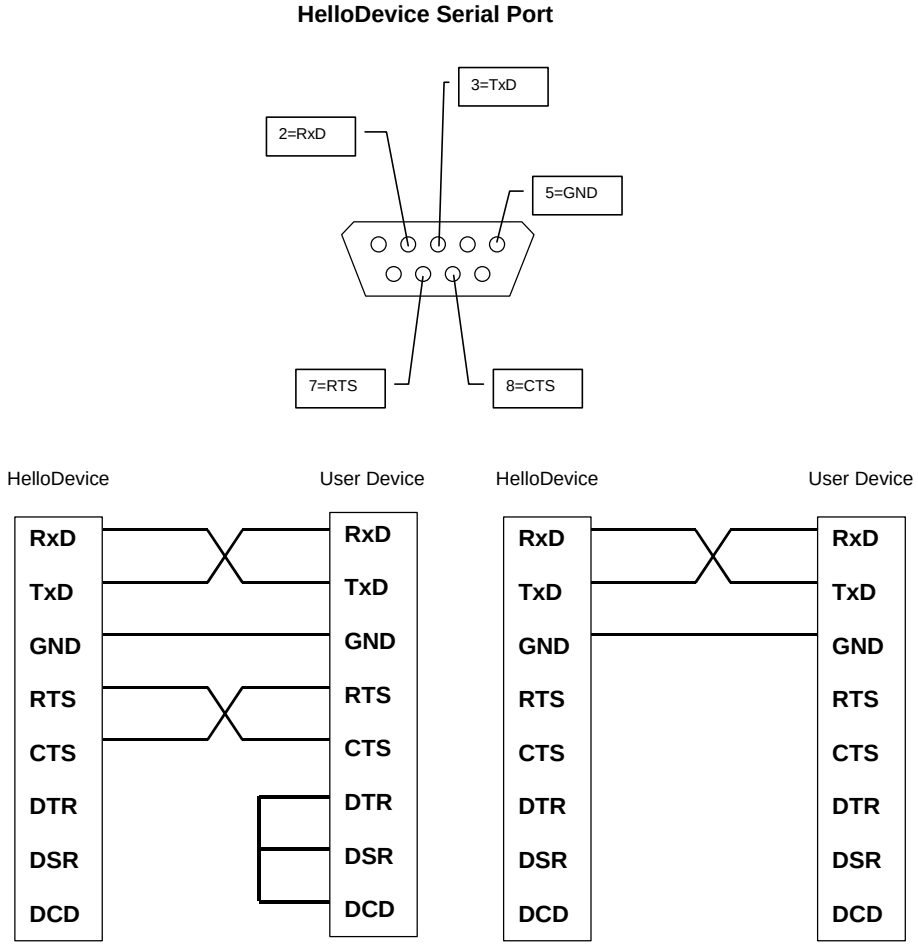


Figure A. HD1320/1320E RS232 Cable Connection Method